


mom

Please type a plus sign (+) inside this box → PTO/SB/05 (11-00)
Approved for use through 10-31-2002. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.**UTILITY
PATENT APPLICATION
TRANSMITTAL**

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No. **X-805-9 US**

First Inventor

Stephen M. Trimberger

Title

Programmable Logic Device with Decryption and Structure for Preventing Design Relocation

Express Mail Label No.

EL539650703US**APPLICATION ELEMENTS**

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO:

**Commissioner for Patents
Box Patent Application
Washington, DC 20231**

1. ☒ Fee Transmittal Form (e.g., PTO/SB/17)
(Submit an original, and a duplicate for fee processing)
2. ☒ Specification [Total Pages **40**]
(preferred arrangement set forth below)
- Descriptive title of the invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to sequent listing, a table, or a computer program listing appendix
- Background of the invention
- Brief Summary of the invention
- Brief Description of the Drawings (if filed)
- Detailed Description
- Claim(s)
- Abstract of the Disclosure
3. ☒ Drawing(s) (35 USC 113) [Total Sheets **16**]
4. Oath or Declaration [Total Pages **2**]
a. ☒ Newly executed (original or copy)
b. ☐ Copy from a prior application (37 CFR §1.63(d))
(for continuation/divisional with Box 16 completed)
☐ **DELETION OF INVENTOR(S)**
Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b)
5. ☐ Application Data Sheet. See 37 CFR 1.76

6. ☐ CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix)
7. Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
a. ☐ Computer Readable Copy
b. Specification Sequence Listing on:
i. ☐ CD-ROM or CD-R (2 copies); or
ii. ☐ paper
c. ☐ Statements verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. ☒ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3.73(b) Statement (when there is an assignee) ☐ Power of Attorney
10. ☐ English Translation Document (if applicable)
11. ☒ Information Disclosure Statement (IDS)/PTO-1449 ☒ Copies of IDS Citations
12. ☐ Preliminary Amendment
13. ☒ Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
14. ☐ Certified Copy of Priority Document(s) (if foreign priority is claimed)
15. ☐ Request and Certification under 35 U.S.C 122 (b)(2)(B). Applicant must attach form PTO/SB/35 or its equivalent.
16. ☐ Other:

17. If a **CONTINUING APPLICATION**, check appropriate box and supply the requisite information below and in a preliminary amendment; or in an Application Data Sheet under 37 CFR 1.76:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No: _____

Prior application information: Examiner: _____ Group / Art Unit: _____

For **CONTINUING** or **DIVISIONAL APPS** only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS☒ Customer Number or Bar Code Label**24309**or ☐ Correspondence address below

Name

Attn: Keith A. Chanroo

Address

City

State

Zip Code

Country

Telephone

408-879-7710

Fax

408-377-6137

Name (Print/Type)

Keith A. Chanroo

Registration No. (Attorney/Agent)

36,480

Signature

Date

November 28, 2000

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

FEE TRANSMITTAL for FY 2001

Patent fees are subject to annual revision

TOTAL AMOUNT OF PAYMENT (\$)**750.00**

Complete if Known

Application Number	Unknown
Filing Date	November 28, 2000
First Named Inventor	Stephen M. Trimberger
Examiner Name	Unknown
Group /Art Unit	Unknown
Attorney Docket No.	X-805-9 US

METHOD OF PAYMENT

1. ☒ The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit
Account
Number

24-0040

Deposit
Account
Name

XILINX, INC.

☐ Charge Any Additional Fee Required
Under 37 CFR §§ 1.16 and 1.17

2. ☐ Payment Enclosed:

☐ Check ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity		Fee Description	Fee Paid
Fee Code	Fee (\$)		
101	710	Utility filing fee	\$710
106	320	Design filing fee	
107	490	Plant filing fee	
108	710	Reissue filing fee	
114	150	Provisional filing fee	
SUBTOTAL (1)		(\$)	710.00

2. EXTRA CLAIM FEES

	Extra	Fee from below	Fee Paid
Total Claims 10 - 20** =	-10	X	\$0
Indep Claims 01 - 3** =	-2	X	\$0
Multiple Dependent		X	

Large Entity		Fee Description	Fee Paid
Fee Code	Fee (\$)		
103	18	Claims in excess of 20	
102	80	Independent claims in excess of 3	
104	270	Multiple dependent claim, if not paid	
109	80	**Reissue independent claims over original patent	
110	18	**Reissue claims in excess of 20 and over original patent	
SUBTOTAL (2)		(\$)	0.00

**or number previously paid, if greater, For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Fee (\$)	Fee Description	Fee Paid
105	130	Surcharge - late filing fee or oath	
127	50	Surcharge - late provisional filing fee or cover sheet.	
147	2,520	For filing a request for reexamination	
112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	Requesting publication of SIR after Examiner action	
115	110	Extension for reply within first month	
116	390	Extension for reply within second month	
117	890	Extension for reply within third month	
118	1,390	Extension for reply within fourth month	
128	1,890	Extension for reply within fifth month	
119	310	Notice of Appeal	
120	310	Filing a brief in support of an appeal	
121	270	Request for oral hearing	
138	1,510	Petition to institute a public use proceeding	
140	110	Petition to revive - unavoidable	
141	1,240	Petition to revive - unintentional	
142	1,240	Utility issue fee (or reissue)	
122	130	Petitions to the Commissioner	
123	50	Petitions related to provisional applications	
126	240	Submission of Information Disclosure Stmt	
581	40	Recording each patent assignment per property (times number of properties)	\$40
146	710	Filing a submission after final rejection (37 CFR 1.129(a))	
149	710	For each additional invention to be examined (37 CFR 1.129(b))	
179	710	Request for Continued Examination (RCE)	
Other fee (specify)			

*Reduced by Basic Filing Fee Paid SUBTOTAL (3) (\$)**40.00**

SUBMITTED BY

Name (Print/Type)	Keith A. Chanroo	Registration No. (Attorney/Agent)	36,480	Telephone	408-879-7710
Signature		Date	11-28-2000		

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

11/28/00

1 PROGRAMMABLE LOGIC DEVICE WITH DECRYPTION AND
2 STRUCTURE FOR PREVENTING DESIGN RELOCATION

3
4 Stephen M. Trimberger
5 Raymond C. Pang
6 Walter N. Sze
7 Jennifer Wong
8

9 FIELD OF THE INVENTION

10 The invention relates to PLDs, more particularly to protection of designs
11 loaded into a PLD through a bitstream.
12

13 BACKGROUND OF THE INVENTION

14 A PLD (programmable logic device) is an integrated circuit structure that
15 performs digital logic functions selected by a designer. PLDs include logic blocks
16 and interconnect lines and typically both the logic blocks and interconnections are
17 programmable. One common type of PLD is an FPGA (field programmable logic
18 device), in which the logic blocks typically include lookup tables and flip flops,
19 and can typically generate and store any function of their input signals. Another
20 type is the CPLD (complex programmable logic device) in which the logic blocks
21 perform the AND function and the OR function and the selection of input signals
22 is programmable.
23

24 Problem with storing bitstream external to PLD

25 Designs implemented in PLDs have become complex, and it often takes
26 months to complete and debug a design to be implemented in a PLD. When the
27 design is going into a system of which the PLD is a part and is to be sold for profit,
28 the designer does not want the result of this design effort to be copied by someone
29 else. The designer often wants to keep the design a trade secret. Many PLDs,
30 particularly FPGAs, use volatile configuration memory that must be loaded from
31 an external device such as a PROM every time the PLD is powered up. Since
32 configuration data is stored external to the PLD and must be transmitted through

EL539650703US

1 a configuration access port, the privacy of the design can easily be violated by an
2 attacker who monitors the data on the configuration access port, e.g. by putting
3 probes on board traces.

4
5 Current solutions and their disadvantages

6 Efforts have been made to encrypt designs, but it is difficult to make the
7 design both secure from attackers and easy to use by legitimate users. The
8 encryption algorithm is not a problem. Several encryption algorithms, for
9 example, the standard Data Encryption Standard (DES) and the more secure
10 Advanced Encryption Standard (AES) algorithm, are known for encrypting blocks
11 of data. The process of cipher block chaining (CBC), in which an unencrypted
12 data word is XORed with the next encrypted data word before decryption allows
13 the DES or AES to encrypt a serial stream of data and these are therefore
14 appropriate for encrypting a bitstream for configuring a PLD. A key used for
15 encrypting the design must somehow be communicated in a secure way between
16 the PLD and the structure that decrypts the design, so the design can be decrypted
17 by the PLD before being used to configure the PLD. Then, once the PLD has been
18 configured using the unencrypted design, the design must continue to be
19 protected from unauthorized discovery.

20 A November 24, 1997 publication by Peter Alfke of Xilinx, Inc. entitled
21 "Configuration Issues: Power-up, Volatility, Security, Battery Back-up" describes
22 several steps that can be taken to protect a design in an existing FPGA device
23 having no particular architectural features within the FPGA to protect the design.
24 Loading design configuration data into the FPGA and then removing the source of
25 the configuration data but using a battery to maintain continuous power to the
26 FPGA while holding the FPGA in a standby non-operational mode is one method.
27 However, power requirements on the battery make this method impractical for
28 large FPGA devices.

29 Nonvolatile configuration memory is another possibility. If the design is
30 loaded at the factory before the device is sold, it is difficult for a purchaser of the

1 For additional security, rather than using nonvolatile memory to preserve
2 keys, the invention preferably uses a battery connected to the PLD to preserve the
3 key when power is removed from the PLD. Whereas it is possible to remove a
4 PLD storing keys in nonvolatile memory, decap the PLD and observe which of the
5 nonvolatile bits are programmed to logic 1 and which are programmed to logic 0,
6 it is believed that it is very difficult to determine the contents of keys stored only
7 in static memory cells since power must be maintained to the memory cells storing
8 the keys in order for the keys to even be stored, and the PLD would have to be
9 decapped, delayered, and probed while operating power is continuous to the
10 PLD.

11
12 Ways an attacker can steal a design once loaded into a PLD

13 If a key does not offer sufficient security, an attacker may break the
14 encryption code and determine the value of the key. The well-known Data
15 Encryption Standard DES used a 56-bit encryption key, and has been broken in a
16 few hours by a sophisticated computer to reveal the key. DES is described by
17 Bruce Schneier in "Applied Cryptography Second Edition: protocols, algorithms,
18 and source code in C" copyright 1996 by Bruce Schneier, published by John Wiley
19 & Sons, Inc., at pages 265-278. If it is desirable to use such a well known
20 encryption standard, then in order to increase security, the configuration data may
21 be encrypted several times using different keys each time, thus strengthening the
22 encryption code by about 2^{56} each time the encryption is repeated. Or it may be
23 encrypted using a first key, decrypted using a second key, and encrypted using a
24 third key, a combination that is part of the triple DES standard. Other encryption
25 algorithms may also be used, and it is not necessary to keep the algorithm secret
26 since the security resides in the key. When the encryption method is symmetrical,
27 the same keys used for encryption are stored in the PLD and used in reverse order
28 for decryption.

29 In a PLD offering multiple keys, if the number of keys to be used and the
30 addresses of all keys were provided in an unencrypted bitstream, an attacker

1 might be able to attack the keys one at a time and more easily determine the key
2 values. To avoid such attack, additional security is achieved by storing within the
3 keys, not the bitstream, an indication of how many keys are to be used and
4 whether a key is the last key of a set or whether more are to follow.

5 If the PLD offers the option of reading back the bitstream after it has been
6 loaded into the PLD, another method that can be used by an attacker is to read
7 back this bitstream. To avoid this method of attacking the design, in one
8 embodiment, a PLD that offers readback and also offers encryption includes the
9 ability to disable the readback feature when encryption has been used. In another
10 embodiment, the PLD that offers the ability to read back encrypts the
11 configuration data before it is read back.

12 Additionally, some PLDs offer the option of partial configuration (where
13 several configuration addresses are specified for loading several portions of a
14 design) and partial reconfiguration (where an existing design is not erased before
15 new design data are loaded). If the PLD offers these options, an attacker could
16 partially reconfigure a PLD to make successive portions of the design visible, and
17 probably learn the whole design. To avoid such an attack, in one embodiment,
18 partial configuration and reconfiguration of PLDs loaded with encrypted designs
19 are disallowed. In another embodiment, several configuration addresses can be
20 specified, but the addresses are encrypted.

21 Yet another mode of attack is to try to flip a bit that indicates the security
22 status of the PLD. Lowering or raising the operating voltage, changing the
23 temperature, and applying noise to certain ports come to mind. To protect against
24 such bit-flipping, when the PLD is operating with a secured bitstream, a secure-
25 mode flag is set, and in one embodiment, if this flag becomes unset, all
26 configuration data is erased. In another embodiment that doesn't allow for
27 reconfiguration while the device is still operating, the configuration data is erased
28 before any bitstream is sent.

29 Another mode of attack is to relocate portions of the encrypted bitstream so
30 that when they are unencrypted they are placed into visible portions of the PLD

not intended by the designer. To prevent this relocation, address information is used in the encryption and decryption processes so that sending a portion of an encrypted bitstream to a different PLD location from that intended by the designer will cause it to decrypt differently into data with no meaning. Cipher block chaining (CBC) is one effective means of achieving this result. In cipher block chaining, the decrypted data packet (block) is combined using the XOR function with the next data block before the next block is decrypted, thus the encrypted data for each data block depends on every block that preceded it and on the order of those blocks. Identical blocks of data will encrypt to different values depending on the value of the data blocks that preceded them. This way, if the order of the blocks is changed, the bitstream will not decrypt correctly because the place where the encrypted bitstream is rearranged will scramble subsequent data. Further, the initial CBC value can be modified to incorporate the address of the data to force the decrypted data to be placed at a specific location in order to decrypt correctly.

Alternatively, if the PLD allowed part of a design to be encrypted and part to be unencrypted, the attacker could add an unencrypted portion to the encrypted portion that would read out information about the encrypted portion of the design. Thus, additional security is achieved by permitting the design to be totally encrypted or totally unencrypted, but not to be mixed. Further to this, in one embodiment, when data are being encrypted, additional security is provided by allowing only a single full-chip configuration following a single starting address for the configuration data.

Further, in order to allow convenient testing and debugging and to allow the PLD manufacturer to communicate freely with its customers (the designers who produce the designs for configuring the PLD), the PLD has both encrypted and unencrypted modes of operating, and when operating in the encrypted mode, parts of the configuration bitstream that control loading of the configuration data into the PLD are still not encrypted.

As another mode of attack, if the PLD manufacturer gives information freely about the configuration bitstream format, including header information and

addresses for loading configuration data, and gives information about the encryption method used, encrypting this well known information would expose the encryption key to possible discovery. Such exposure is avoided by encrypting only the actual configuration data and leaving control information unencrypted.

If the PLD manufacturer allows the key memory to be used in both secure and non-secure modes, an attacker could simply learn the keys by placing the key memory into non-secure mode and reading out the keys. To avoid such attack, the PLD manufacturer includes a circuit that causes all keys plus any configuration data loaded into the PLD to be erased when the key memory is moved to non-secure mode.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows functional relationships in a prior art FPGA.

Fig. 2a, 2b, 2c, and 2d show bitstream format and commands that can be included in a prior art bitstream.

Fig. 3 shows functional relationships in an FPGA according to one embodiment of the present invention.

Fig. 4a, 4b, 4c, and 4d show bitstream format and commands that can be included in a bitstream of the present invention.

Fig. 5a and 5b show example unencrypted and encrypted bitstreams.

Fig. 6 shows configuration logic 29 and the lines in bus 27 and bus 28 leading to decryptor 24.

Fig. 7a shows the modified starting value for outer cipher block chaining with triple encryption used in one embodiment of the invention.

Fig. 7b shows the corresponding starting value and decryption process used with Fig. 7a.

Fig. 8 shows flow of the operations for processing a bitstream.

Fig. 9 shows a state machine implemented by decryptor 24 to evaluate key order.

Fig. 10a shows the structure of key memory 23 of Fig. 3.

Fig. 10b shows the structure of the memory cells of Fig. 10a.

Fig. 11 shows the steps performed by control logic 23a of Fig. 10a to erase keys when made non-secure.

Fig. 12 shows in more detail the battery supply switch of Fig. 10a.

Figs. 13 and 14 show the level shift circuit and voltage detection circuit of the battery supply switch of Fig. 12.

Fig. 15 shows a state machine for erasing a design when a secure mode is exited.

Fig. 16 shows a block diagram of elements for loading configuration memory and reading back configuration, including lines disabled when encryption is present.

DETAILED DESCRIPTION

Fig. 1 shows a prior art structure for an FPGA 10. The FPGA includes programmable logic 11, typically comprising (1) logic blocks with lookup table combinatorial logic function generators, flip flops for storing lookup table outputs and other values, and multiplexers and logic gates for enhancing the logic ability of the programmable logic (2) routing lines and programmable interconnection points for routing signals around the FPGA, and (3) input/output blocks for driving signals between the routing lines and the external pins of the FPGA.

The FPGA also includes configuration memory 12 for turning on routing transistors, controlling multiplexers, storing lookup tables and controlling the input/output blocks, all of this for the purpose of configuring the FPGA to perform the function desired by the designer(s). Bus 16 connects configuration memory 12 to programmable logic 11 and is typically a distributed set of control lines located throughout the FPGA. Some Xilinx products (e.g. XC6200) have included a bus 17 by which programmable logic 11 causes configuration logic 14 to send programming information to configuration memory 12. Such a structure is described by Kean in U.S. Patent 5,705,938.

1 FPGA 10 further includes a JTAG logic block 13 for interfacing with JTAG
2 port 20, especially intended for testing of the board in which the FPGA will be
3 placed. JTAG logic block 13 implements the IEEE standard 1532, which is a
4 superset of the IEEE standard 1149.1. JTAG allows debugging of a design at the
5 board level.

6 Finally FPGA 10 includes configuration logic 14 for responding to a
7 configuration bitstream from external source 15 on configuration access port
8 21 and for interfacing with JTAG logic block 13. The bitstream on configuration
9 access port 21 is treated as words, in one embodiment 32-bit words. Several of the
10 words, usually at or near the beginning of the bitstream, are used for setting up
11 the configuration process and include, for example, length of a configuration
12 memory frame, and starting address for the configuration data. Bus 19 allows
13 communication between configuration logic 14 and JTAG logic block 13 so that the
14 JTAG port can be used as another configuration access port. Bus 18 allows
15 communication between configuration logic block 14 and configuration memory
16 12. In particular, it carries addresses to select configuration frames in memory 12,
17 control signals to perform write and read operations, and data for loading into
18 configuration memory 12 or reading back from configuration memory 12.

19 Configuration Logic block 14 receives instructions and data, and processes
20 the data according to the instructions. These instructions come into configuration
21 logic 14 as a bitstream. An instruction, or header, is usually followed by data to
22 be acted upon. Fig. 2a shows an example bitstream structure. Header A specifies
23 an action and specifies that a single word, Data A, will follow. Header B specifies
24 an action and in this case specifies that 4 words of data will follow to be acted
25 upon.

26 Fig. 2b shows the default format (format type 001) for a 32-bit header word
27 in the bitstream used in the Virtex(R) devices available from Xilinx, Inc. (Virtex is a
28 registered trademark of Xilinx, Inc., assignee of the present invention). This
29 format includes three bits to indicate the format type (001), two bits to specify an
30 op code, 16 bits for a configuration logic register address, and 11 bits for a word

count. The op code can designate a read operation, a write operation, or no operation. For example, 00 can designate no operation, 01 can designate read and 10 can designate write. The 11 bits for word count can specify 2^{11} words or 2048 words. As shown in Fig. 2c, if the word count is greater than this, the word count bits in format type 001 are set to 00000000000 and the header of format type 001 is followed by a header of format type 2. Format type 2 uses 27 bits to specify word count, and can thus specify 2^{27} words or 2.68 million words.

Fig. 2d shows the kinds of control information that can be loaded into the registers of Configuration Logic 14 by headers for a Virtex bitstream. For example, a header (of format 001) having the configuration logic register address 0000 specifies that the next 32-bit data word should be loaded into the cyclic redundancy check (CRC) register. (Virtex devices use a 16-bit cyclic redundancy check value so some bits will be padded with 0's.) If the header includes an address 0001, the next data will be loaded into the Frame Address register in order to specify a frame (column) in configuration memory 12 to receive or provide data.

The Configuration Logic Register address (16 bits) shown in Fig. 2b provides the 4-bit values shown in the left column of Fig. 2d that select one of the registers in configuration logic 14 (Fig. 1) into which to place the next 32-bit data word. The Frame Length register (address 1011) specifies the length of the frame into which the configuration data will be loaded. (Frame length, or column height, depends upon the size of the PLD. Larger PLDs usually have taller columns or longer frames. Specifying the frame length in the bitstream and storing the frame length in a register rather than providing a different structure in the PLD for placing the data words into frames allows the internal configuration logic to be identical for PLDs of different sizes.)

For readback, a read command is placed in the op code field and the Frame Data Output register is addressed, followed by a Word Count (using Command Header Format 2 if necessary). The specified number of words is read back from configuration memory 12, starting at the address specified in the Frame Address

1 register, and shifted out on either configuration access port 21 or JTAG port 20.
2 (Readback data is returned to the port that issued the readback instruction) .

3 Specifying a word count in a bitstream header or pair of headers (Figs. 2b
4 and 2c) sets a counter that counts down as the data words are loaded. For many
5 configuration logic register addresses the word count is 1. But if the bitstream
6 header has a configuration logic address of 0010 or 0011 to indicate configuration
7 data are being loaded in or read back, the word count will be much larger. This is
8 when header format 2 of Fig. 2c is used. Data loaded into configuration memory
9 12 through the frame data input register (address 0010) or read out through the
10 frame data output register (address 0011) is called the design data because it
11 causes the FPGA to implement a design or shows the status of a design. The other
12 register data are control data since they control how the configuration logic
13 behaves while the logic is being configured or read back.

14 Further detail about configuration of Virtex devices can be found in the
15 "Virtex Configuration Guide" published October 9, 2000 by Xilinx, Inc. (assignee
16 of the present invention), 2100 Logic Drive, San Jose, CA 95124. This
17 configuration guide is incorporated herein by reference.

18 Configuration logic 14 typically performs a cyclic redundancy check on a
19 configuration bitstream coming in (see Erickson, U.S. Patent 5,321,704
20 incorporated herein by reference, or see pages 39 through 40 of the above
21 referenced Virtex Configuration Guide), reads header bits indicating the frame
22 length of the part being configured and the word count of the configuration data,
23 reads address instructions identifying where to load configuration data, collects
24 frames of configuration data and loads them into columns of configuration
25 memory 12 indicated in the addresses. Configuration logic 14 also controls
26 readback of configuration data and flip flop values from configuration memory 12
27 to an external location. In a Virtex FPGA available from Xilinx, Inc., readback can
28 be done through either JTAG port 20 or through configuration access port 21.

29 Configuration logic 14 can also receive configuration data from
30 programmable logic 11. More information about prior art FPGA structures in

which part of the FPGA configures another part of the FPGA can be found in Kean, U.S. Patent 5,705,938. More information about architectures of FPGAs similar to the Virtex architecture can be found in Young et al., U.S. Patent 5,914,616. Both patents are incorporated herein by reference. The format of a bitstream used with the Virtex product available from Xilinx, Inc., assignee of the present invention, is described in an Application Note, XAPP138, entitled "Virtex FPGA Series Configuration and Readback" available from Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 published Oct. 4, 2000.

PLD with Decryption

Fig. 3 shows a block diagram of an FPGA (a type of PLD) according to one embodiment of the present invention. Some elements are the same as shown in Fig. 1, are given the same reference numbers, and not explained again. In addition, Fig. 3 includes an expanded configuration logic unit 29, a decryptor 24 and a key memory 23. Fig. 3 shows an embodiment in which key memory 23 is loaded on bus 25 from JTAG access port 20. In other embodiments, key memory 23 is loaded through another port. Bus 25 carries data, addresses, and control signals to perform write and read operations and allows programming of the decryption keys from JTAG port 20. In one embodiment, bus 26 allows programming of the keys from the configuration port. In another embodiment, bus 26 is eliminated. In yet another embodiment, bus 26 is present and bus 25 is eliminated. In an embodiment described further herein, bus 26 carries security data from key memory 23 to configuration logic 29. In one embodiment, bus 27 carries encrypted configuration data from configuration logic 29 to decryptor 24 and carries decrypted configuration data back to configuration logic 29. Bus 28 allows decryptor 24 to access the keys for decrypting data. When the structure of Fig. 3 is being loaded with encrypted data, an attacker who monitors the bitstream as it is being loaded receives only the encrypted bitstream and can not learn the user's design by this method.

Partially Encrypted Bitstream

According to another aspect of the invention, the bitstream comprises two portions, a data portion representing the user's design that can be encrypted or not, and a control portion controlling loading of the bitstream (for example giving addresses of columns in the PLD into which successive portions of the bitstream are to be loaded, providing a cyclic redundancy check (CRC) code for checking reliability of the loading operation, and a starter number for cipher block chaining (CBC), a technique that prevents a "dictionary attack" where the decrypted data can be deduced from the frequency of occurrence of the encrypted data). In a preferred embodiment of the invention, the data portion may be encrypted but the control portion is unencrypted. This provides additional security because the PLD manufacturer needs to describe freely the control features of the bitstream, and if this relatively well known control information were encrypted, an attacker might be able to decrypt this information and use this information to decrypt the entire bitstream. Further, keeping the control portion of the bitstream unencrypted makes it easier for the PLD to use the information.

In another embodiment, used when the order of addresses in which configuration data is loaded may be useful to an attacker in analyzing the design, the address of the configuration data is also encrypted, but other control information in the configuration bitstream remains unencrypted.

Bitstream Format

Figs. 4a-4d illustrate differences in bitstream format and registers of configuration logic 29 in comparison to the format and registers of configuration logic 14 of the prior art product shown in Figs. 2a-2d. As shown in Fig. 4a, the bitstream still includes header words followed by data words. In a typical configuration, several control data words will be loaded into registers before encrypted configuration data begins. Fig. 4a shows an example in which three header words Header A, Header B, and Header C are each followed by three unencrypted control data words Data A, Data B, and Data C. (In an actual

configuration, more than three control data words will likely be provided.) Next, Header D specifies that encrypted configuration data will follow and is followed by multiple words Data 1D, Data 2D, Data 3D, etc. of encrypted configuration data. These words have been shaded in Fig. 4a to emphasize that this data is encrypted.

As shown in Figs. 4b and 4c, a fourth op code has been added. In addition to the values 00 for no operation, 01 and 10 for read and write without decryption, the new value 11 specifies that writing is to be with decryption. (It is not important what code or what method is used to specify that decryption is to be used or even that it is specified through an op code. It is just important that optional encryption and decryption be allowed and indicated, so that designers can make use of this option. In the embodiment of Fig. 4d, two new configuration logic registers are added. Shown at addresses 1100 and 1101 are the register for holding a cipher block chaining (CBC) starter value and the address for the initial encryption key.

Optional Encryption

According to another aspect of the invention, a PLD can accept both encrypted and unencrypted data portions of the bitstream. The control portion of the bitstream indicates whether the data portion of the bitstream is encrypted. If the data portion of the bitstream is encrypted, it is diverted within the PLD to a decryptor and after decryption is used to configure the PLD. If unencrypted, it is not diverted, and is used directly to configure the PLD.

There are some occasions for which it is preferable not to encrypt the bitstream. Certain test activities used during debugging a design require reading back the configuration information. It is more straight forward to diagnose a configuration problem if an encryption step has not been performed (especially if the designer is trying to determine whether encryption has anything to do with the problem). Also, if several designers are writing code to be implemented in parts of the PLD and different parts of the PLD are to be configured at different

1 times, it may be necessary to make all portions of the bitstream visible, and to
2 allow the PLD to be partly reconfigured.

3 Figs. 5a and 5b show example bitstream portions representing the same
4 design, first unencrypted and then encrypted, to illustrate the differences between
5 an unencrypted bitstream and an encrypted bitstream in one embodiment of the
6 invention. An actual bitstream includes the 0's and 1's at the right of the figures
7 and none of the text at the left. The text at the left is provided to explain the
8 meaning of the bits to the right. These bitstream portions use the commands
9 illustrated in Figs. 4b-4d. In order to emphasize the differences between the
10 unencrypted version of Fig. 5a and the encrypted version of Fig. 5b, the
11 differences are shown in bold.

12 Looking at Fig. 5a, after a dummy word (a constant high signal interpreted
13 as all 1's) and a sync word with a specified pattern of 1's and 0's, the next word is
14 of type 001 with an op code of 10, has an address of 0000000000010000 and a word
15 count of 00000000001. Thus this word addresses the command register CMD and
16 specifies that one word will be written there. Fig. 5a has been annotated to the left
17 of the bitstream to indicate that this word is Type 1 and indicates to write 1 word
18 to CMD. The following word 111 is the data to be placed in command register
19 CMD, and resets a CRC (cyclic redundancy check) register. (In a preferred
20 embodiment, the PLD includes a circuit, not shown, such as described by
21 Erickson in U.S. Patent 5,598,424 to calculate a CRC value from the bitstream as
22 the bitstream is being loaded, and protects against glitches in the bitstream
23 voltages that might cause incorrect bits to be loaded.) Next, a header word
24 specifies that the format is again type 1 and it specifies to write 1 word to the
25 frame length register FLR. The data word that follows, 11001, specifies the frame
26 length (25 words). Similarly, several additional header and data words follow,
27 including the header specifying the word to be written to the frame address
28 register FAR. In this case, the following data word indicates data will start at
29 address 0. Finally, after these registers have been loaded, a command comes to
30 write data to the frame data input register FDRI, and since quite a bit of data will

1 be written, the word count is given as 000000000000 and a header of type 2 specifies
2 that 10530 words will be written to the FDRI register. This is the actual design
3 data that causes the PLD to be configured. Thus the next 10530 words in the
4 bitstream are design data. Finally, to assure that data have been loaded correctly,
5 the CRC value calculated by the device that originated the configuration data is
6 loaded and compared to the CRC value that has been calculated by the PLD.

7 Additional commands and data are loaded in order to indicate that configuration
8 is complete and to move the PLD into operation mode.

9 Fig. 5b is similar to Fig. 5a, and differs only where the data and annotations
10 are shown in bold. In Fig. 5b, the data are encrypted, and additional commands
11 are used to provide the initial key address and to write two words (64 bits) to the
12 CBC (cipher block chaining) register. Next, a type 1 header includes the op code
13 11 and indicates that data will be decrypted before being written to frame data
14 input register FDRI. A type 2 header follows, again with the op code 11, giving
15 the instruction that 10530 words are to be decrypted and written to data input
16 register FDRI. The 10530 encrypted data words then follow. Then the CRC word
17 follows for confirming that the (encrypted) data were loaded correctly. Finally,
18 the additional commands and data are sent, and place the PLD into operation
19 mode if all is correct.

20 Decryption Process

22 Fig. 6 shows how optional decryption is accomplished in one embodiment.
23 Fig. 6 shows the detail of configuration logic 29 and of buses 27 and 28 leading
24 into decryptor 24. Bus 27 includes the following:

- 25 •the 3-bit initial decryption key address "Init_key_addr" taken from
26 register address 1101 (Fig. 4d) in configuration logic 29,
- 27 •the 64-bit modified cipher block chaining value "modCBC". This value is
28 formed by replacing the lower order bits of the 64-bit CBC value
29 taken from register address 1100 (Fig. 4d) in configuration logic 29
30 with the 22-bit Frame Address value specified in Register 0001.

- the 64 lines "Encrypted_data" for loading encrypted data, taken from the bitstream,
- the 64 lines "Decrypted_data" for returning the decrypted data produced by decryptor 24 to configuration logic 29,
- a line for the signal "Enc_data_rdy" that tells decryptor 24 that data is on the "Encrypted_data_ lines and that decryptor 24 can start decrypting,
- a line for the signal "Dec_data_rdy" that tells configuration logic 29 that decryption on a 64-bit word is complete and is available on the "Decrypted_data" lines, and
- a Bad_key_set line used by decryptor 24 to cause configuration logic 29 to abort the configuration and set a status register accordingly when the keys have not been used as specified, for example, by the bits in key memory that designate whether the keys are to be first, middle, or last of a set. In the embodiment shown in Fig. 4d, the status register is at address 0111, and the Bad_key_set error is indicated by storing a logic 1 in one of the bits.

Bus 28 is comprised of the following:

- 3 lines for the key address, which is initially the key address provided from bus 27, but which is updated each time a new key is used,
- 56 lines for the decryption key, and
- 2 lines for indicating whether the decryption key is the first, middle, last, or only key to be used.

Preventing Design Relocation

One potential attack on a design in an encrypted bitstream is to change the frame address register (starting address) in the encrypted bitstream so that when it is decrypted it is loaded into a portion of the FPGA visible when the FPGA is being used. In some designs the content of the block RAM is visible. In all designs the configuration of the input/output ports is visible and therefore the

configuration bits can be determined. Thus if successive portions of the design were moved to visible portions of the FPGA, even though the FPGA did not function properly, an attacker could in repeated relocation learn the contents of the unencrypted bitstream.

To prevent design relocation, in one embodiment, an initial value used by the cipher block chaining method used with the DES standard is modified. Figs. 7a and 7b show the encryption and decryption portions of a triple DES algorithm, respectively, as modified according to the invention. The standard cipher block chaining method starts the encryption process by XORing a starting number (can be designer supplied or randomly generated) with the first word of data to be encrypted. According to the invention, part of the random number is replaced by address information, in the present example the 22-bit address of the first frame into which data will be loaded in configuration memory 12. The starter CBC value, a 64-bit number, has its least significant bits, labeled x, replaced by the frame address, labeled y, to produce a modified 64-bit value that depends upon the address into which data will be loaded. This modified CBC value is XORed with the first word of configuration information Word1. Then the encryption algorithm is used to produce the first encrypted word Encrypted Word1, which is placed into the bitstream. Fig. 7a shows a triple encryption algorithm with outer cipher block chaining, comprising an encryption step enc_1 using the first key, followed by a decryption step dec_2 using the second key, followed by an encryption step enc_3 using the third key. This first encrypted word Encrypted Word1 is XORed with the second unencrypted word Word2 and the encryption process is repeated to produce encrypted Word2. The XOR chaining continues until all configuration data have been encrypted.

As shown in Fig. 7b, the PLD must perform the reverse process to derive the decrypted words. For the above encryption sequence, the decryption sequence would be decryption step dec_1 using key 3, then encryption step enc_2 using key 2, then decryption step dec_3 using key 1. Importantly, part of the initial value for generating Decrypted Word1 is to use the same frame address for both encryption

1 and decryption. The PLD, not the bitstream, generates the modified CBC value
2 from the frame address stored in the frame address register, which is also used to
3 specify the frame of configuration memory 12 into which configuration data are to
4 be loaded. So if an attacker changes the frame address into which the data are to
5 be loaded, the modified CBC value changes accordingly, and the configuration
6 data are not correctly decrypted.

7 The XOR step produces the original data that was in the designer's bitstream
8 before it was encrypted. Original Word1 = Decrypted Word1, for example. This
9 decrypted configuration data is sent on bus 27 (Fig. 3) to configuration logic 29.

11 Configuration Logic 29

12 Configuration logic 29 includes the structures to support optional encryption
13 as well as the structures to prevent design relocation and a single key attack. As
14 shown in Fig. 6, configuration logic 29 includes a holding register 292, control
15 logic 291, configuration registers (FDRI, FAR, CRC, and init CBC are shown),
16 decryptor 24 interface multiplexers 294 and 295, 64-bit assembly register 297, and
17 registers 298 and 299 (for interfacing with configuration access port 21). A 64-bit
18 shift register 299 receives data from configuration access port 21, which can be a
19 single pin for 1-bit wide data or 8 pins for 8-bit wide data. This data is loaded into
20 64-bit shift register 299 until register 299 is full. Then these 64 bits are preferably
21 shifted in parallel into 64-bit transfer register 298. From there, multiplexer 296b
22 alternately selects right and left 32-bit words, and multiplexer 296a moves the data
23 32 bits at a time either into holding register 292 or alternately into High and Low
24 portions of assembly register 297 as controlled by control line M. When loading of
25 the bitstream begins, line M and a clock signal not shown cause multiplexers 296a
26 and 296b to move data from 64-bit transfer register 298 to holding register 292.
27 From there these words are applied to control logic 291. If the word is a header,
28 control logic 291 interprets the word. If the op code indicates the data to follow
29 are to be written unencrypted, control logic 291 places an address on bus G to
30 select a register, places a signal on line L to cause multiplexer 294 to connect bus B

1 to bus D, and applies the following word on bus B. On the next clock signal (clock
2 signals are not shown), the data on bus D are loaded into the addressed register.
3 All registers shown in Fig. 4d can be loaded this way. The init CBC register for
4 loading the initial cipher block chaining value is a 64-bit register and receives two
5 consecutive 32-bit words, as shown in Fig. 5b and discussed above.

6 A modified CBC value formed from (1) the original CBC value stored in the
7 init CBC register and (2) the initial frame address stored in the FAR register is
8 available to decryptor 24. In one embodiment, the initial frame address in the
9 FAR register uses no more than 32 bits while the init CBC value uses 64 bits. In
10 the embodiment of Fig. 6, the 64-bit bus providing the modified CBC value
11 includes 22 bits from the frame address register FAR and 42 bits from the init CBC
12 register. Important to the security provided by the present invention, note that
13 this value depends upon where configuration data will be loaded. If an attacker
14 were to try to load encrypted data into a different place by changing the contents
15 of the FAR register, the modCBC value fed to decryptor 24 would also change.

16 When the op code command to decrypt a number of words of configuration
17 data is received by control logic 291, the decryption process begins. Control line
18 M causes multiplexer 296a to apply data from transfer register 298 to bus A
19 leading to assembly register 297. Control bus H alternately connects bus A to the
20 High[31:0] and Low[31:0] portions of encrypted data register 297 to form a 64-bit
21 word to be decrypted. Control logic 291 then asserts the Enc_data_rdy signal,
22 which causes decryptor 24 to decrypt the data in register 297.

23 To perform the decryption, decryptor 24 applies a key address KeyAddr on
24 bus 28 to key memory 23 (Fig. 3). This causes key memory 23 to return the 56-bit
25 key in that address on the 56-bit Key lines. It also causes key memory 23 to return
26 two additional bits "Order" also stored in the key data at that address. For the
27 first decryption key, these two bits must indicate that this is a first key or an only
28 key. If not, decryptor 24 asserts the Bad_key_set signal, which causes control logic
29 29 to abort the configuration operation. If these two bits indicate the key is a first
30 or only key, decryptor 24 performs the decryption, using for example the well

1 known DES algorithm (described by Schneier, *ibid*). If the key isn't an only key,
2 decryptor 24 then gets the key at the next address in key memory 23, and checks
3 to see if the two Order bits indicate it is a middle or last key. If not, the
4 Bad_key_set signal is asserted and the configuration is aborted. If so, decryption
5 is performed. If it is a middle key, another round of decryption is done. If it is the
6 last key, decryptor 24 forms the XOR function of the decrypted word and the
7 value modCBC. Decryptor 24 then places the resultant value on the 64-bit
8 Decrypted_data bus and asserts the Dec_data_rdy signal. This causes control logic
9 291 to place signals on control line K to cause multiplexer 295 to break the 64-bit
10 word into two sequential 32-bit words. Control logic 291 places a signal on line L
11 to cause multiplexer 294 to forward the 32-bit words of decrypted data to bus D.
12 Control logic 291 also places address signals on bus G to address frame data input
13 register FDRI. The next clock signal moves the decrypted data to bus E where it is
14 loaded into the frame register and when the frame register is full, eventually
15 shifted into configuration memory 12 at the address indicated in the FAR register.

16 The modCBC value is used only once in the decryption operation.
17 Subsequent 64-bit words of encrypted data are decrypted and then chained using
18 the previously decrypted data for the XOR operation. (The value stored in the
19 FAR register is also used only once to select a frame address. Subsequently, the
20 frame address is simply incremented every time a frame is filled.)

21

22 Flow of Operations

23 Fig. 8 indicates the flow of operations performed by configuration logic 29
24 and decryptor 24. Configuration logic 29 begins at step 70 by loading the bitstream
25 headers and placing the corresponding data into configuration logic registers
26 shown in Fig. 4b, including determining bitstream length. At step 71, as a further
27 part of the start-up sequence, configuration logic 29 reads the first configuration
28 memory address. Recall that the bitstream format includes an op code that
29 indicates whether encryption is being used. Step 72 branches on the op code
30 value. If encryption is not used, the process is shown on the left portion of Fig. 8.

1 If encryption is used, the process is shown in the right of Fig. 8. For no encryption,
2 at step 73, configuration logic 29 sets a counter equal to the bitstream word count
3 (see Fig. 4c). At step 74, 32 bits (1 word) of configuration data are sent to the
4 addressed frame of configuration memory 12. If step 75 indicates the counter is
5 not finished, then at step 76 the counter is decremented and the next 1 word of
6 configuration data are sent to configuration memory 12. When the counter has
7 finished, configuration logic 29 performs cleanup activities including reading the
8 final cyclic redundancy value to compare with a value at the end of the bitstream
9 to determine whether there were any errors in loading the bitstream.

10 If step 72 indicates the bitstream is encrypted, the counter is loaded with the
11 word count, and at step 81 the process loads the initial key address from key
12 address register 293 (Fig. 6) into decryptor 24.

13 At step 82, two words (64 bits) of encrypted configuration data are loaded
14 into decryptor 24. At step 83 the addressed key is loaded into decryptor 24. In
15 one embodiment, a 64-bit number is loaded into decryptor 24. This 64-bit number
16 includes a 56-bit key, two bits that indicate whether it is the first, middle, last, or
17 only key, and some other bits that may be unused, used for parity, or used for
18 another purpose. In another embodiment, the 64-bit key data includes a single bit
19 that indicates whether it is or is not the last key. In yet another embodiment, the
20 64-bit key data includes an address for the next key so the keys don't need to be
21 used in sequential order. In another embodiment, extra bits are not present and
22 the key data uses less than 64 bits. In yet another embodiment, the bitstream
23 rather than the key indicates how many keys are to be used, but this is believed to
24 be less secure because an attacker can see how many keys are used and perform a
25 single key attack, breaking one key at a time, whereas using the keys to indicate
26 how many keys are to be used does not give this information to an attacker.

27 At step 84, decryptor 24 decrypts the 64-bit data with the 56-bit key using,
28 for example, the DES algorithm. The DES algorithm is described in the above-
29 mentioned book by Bruce Schneier at pages 265 to 278. Other encryption
30 algorithms may also be used, for example, the advanced encryption standard AES.

1 Other algorithms may require more key bits. For example AES requires a key of
2 128 to 256 bits.

3 Step 85 determines whether more keys are to be used. The two bits that
4 indicate whether the key is first, middle, last, or only key are examined to
5 determine whether this is the last key, and if not, the key address is incremented
6 and decryptor 24 addresses the next key in memory 23.

7 After the last key has been used, at step 87, the modified CBC value shown in
8 Fig. 6 as a 64-bit value from combining registers FAR and init CBC is XORed with
9 the decrypted value obtained in step 87. In one embodiment, 22 bits of the 64-bit
10 random number loaded into the CBC register are replaced with the frame address
11 of the beginning of the bitstream. The goal of the encryption process is to have
12 every digit of the 64-bit encrypted value be a function of all previous bits plus the
13 key. The goal of combining the CBC value with the first address is to cause the
14 decrypted values to change if the bitstream is loaded into a different address from
15 the intended starting address. Step 87 achieves both goals. The new CBC value is
16 then stored. Storage may be in the FAR and init CBC registers shown in Fig. 6, or
17 in another register located in decryptor 24.

18 At step 88, this decrypted configuration data is sent on bus 27 (Fig. 3) to
19 configuration logic 29. Configuration logic 29 calculates an updated cyclic
20 redundancy check value to be compared with the cyclic redundancy value stored
21 in the CRC register at the end of the loading process. If configuration logic 29 has
22 been set to use encryption, a multiplexer in configuration logic 29 forwards this
23 decrypted configuration data to the addressed column of configuration memory
24 12.

25 At step 89 the counter is checked and if not finished, at step 96 the counter is
26 decremented and the process returns to step 82 where the next 64 bits (2 words)
27 are loaded from the bitstream.

28 Finally, when step 89 indicates the counter is finished, at step 90, a CRC
29 (cyclic redundancy check) value in the bitstream is compared with a CRC value
30 calculated as the bitstream is loaded. If the values agree, configuration is complete

1 and the FPGA goes into operation. If the values do not agree, a loading error has
2 occurred and the entire configuration process is aborted.

3 4 Evaluating Key Order - Preventing Single Key Attack

5 Fig. 9 shows a state machine implemented by decryptor 24 to evaluate key
6 order. The state machine remains in state S1 until the Enc_data_ready signal is
7 activated. This signal indicates decryption can begin and moves to decision state
8 Q1 where decryptor 24 applies the address specified by the address Init_key_addr
9 on bus 27 to bus 28, reads back a key and a key order, and from the two bits of key
10 order data determines whether the key is a first or only key. If not, decryptor 24
11 sends the Bad_key_set signal to control logic 291 and causes configuration logic 29
12 to abort the configuration. If the address is first or only, decryptor 24 goes to state
13 S3, which decrypts the data. Then the state machine goes to decision state Q2,
14 which determines whether the key is last or only. If so, decryption is complete
15 and at state S4 decryptor 24 returns the decrypted data to configuration logic 29.
16 If not, in state S5, decryptor 24 increments the key address, and gets the new key.
17 The state machine asks question Q3 to determine whether the next key is a middle
18 or last key. If not, state S2 causes the configuration to abort. If the key is middle
19 or last, the state machine returns to state S3 to decrypt the data again. In another
20 embodiment, in state S4 decryptor 24 also performs the step of XORing the
21 decrypted data with a CBC value.

22 The benefit of storing the key order within the keys is that an attacker can
23 not implement a single key attack because the attacker can not prevent decryptor
24 24 from using all the keys specified by key memory 23 (as intended by the
25 designer) when performing decryption. It is not necessary to ask the second and
26 third questions Q2 and Q3 to protect against an attacker using a single key attack,
27 since the key order is stored within the key data inside the PLD. However, it is
28 beneficial to the designer or board tester who loads the keys to ask all three
29 questions to make sure that each key has been labeled correctly when it is loaded.

1 In one embodiment, decryptor 24 uses the triple DES standard with a
2 decryption-encryption-decryption sequence, alternating the algorithm (only
3 slightly) each time another key is used. Such a combination is in accordance with
4 the ANSI X9.52 1998 Triple DES standard. In another embodiment, decryption is
5 used each time.

6 7 Key Memory 23

8 The circuit shown in Fig. 10a includes three components: battery supply
9 switch 22, control logic 23a and key registers 23b. Control logic circuit 23a and
10 key registers 23b comprise key memory 23 of Fig. 3. In the embodiment of Fig.
11 10a, key registers 23b comprise six 64-bit words. Of course, other key memory
12 sizes may alternatively be used. In other embodiments, there may be far more
13 than six keys stored in key memory 23, and more than 3 bits needed to give the
14 address of the key to be used. The power supply for key registers 23b comes from
15 battery supply switch 22 on line VSWITCH. When key memory supply voltage
16 VCCI is insufficient or not present, battery supply switch 22 applies the battery
17 backup voltage VBATT to the VSWITCH line so that VSWITCH carries a positive
18 voltage.

19 In this embodiment each key register has 64 memory cells. Each cell receives
20 a write enable signal WE, that when high causes data to be written to the cell and
21 when low causes data in the cell to be held. Cells in one register have a common
22 write enable signal WE. When the PLD supply voltage (different from VCCI) is
23 absent such that the WE signals are not actively driven, weak pull-down
24 transistors such as T1 pull down the WE signal so that none of the key memory
25 registers can be addressed, and none of the memory cells are disturbed.

26 In one embodiment, the JTAG port of a PLD is used to load decryption keys
27 into the PLD. The memory cell supply voltage is at the device voltage level of
28 VCCI during normal operation, and in one embodiment this level is between 3.0
29 and 3.6 volts. Signals applied to the JTAG port may be several different voltages.
30 Also, there may be several different internal voltages. Thus voltage translation is

needed. This voltage translation is performed in the memory cells. Detail of a memory cell is shown in Fig. 10b. The latch comprising inverters I1 and I2 is powered by VSWITCH and is thus powered whether or not a device supply voltage VCCI is present. The WE signal and the inverted data signal data_b both operate at the 1.5 volt level. These signals drive NMOS transistors T4, T5, and T6, and through inverter I3 (also using the 1.5 volt supply voltage) transistor T7. Fig. 10b shows that when WE is low, transistors T4 and T5 are off, and the content of the latch comprising inverters I1 and I2 is retained. When WE is high, one of inverters I1 and I2 is pulled low, thus loading the new data into the latch.

Control logic circuit 23a receives signals from JTAG bus 25 (also shown in Fig. 3). JTAG bus 25 includes control signals for writing, reading, setting the secure mode, and data and address buses. This interface conforms to the IEEE 1532 JTAG standard. Before key memory 23 can be accessed through JTAG bus 25, the security status (bus 26) is placed in non-secure mode, which can be done using the ISC_PROGRAM_SECURITY instruction (see Fig. 10a) and applying logic 1 to bit 0 of the key data bus. Key memory 23 is written to and read (for verification) from JTAG bus 25 using the ISC_PROGRAM and ISC_READ instructions of the IEEE 1532 standard. Control logic 23a includes a decoder for decoding the 3-bit address signal ADDR from JTAG bus 25 to produce a low-going pulse on the addressed one of write strobe lines ws_b[5:0] if the ISC_PROGRAM instruction appears on JTAG bus 25, or a high signal on the addressed one of read select lines rsel[5:0] if the ISC_READ instruction appears on JTAG bus 25. One of the six 64-bit words can be read by applying a high signal to one of the six read select lines rsel[5:0], which causes read multiplexer 23d to place the selected word on the 64 output lines q[63:0]. Only one of the write select lines or read select lines is selected at one time. When no read select signal is asserted, a high park_low signal causes 64 transistors 23e to pull down the 64 lines q[63:0] and prevent these lines from floating.

If key memory 23 is operating in non-secure mode, the 64-bit words can be read from key registers 23b to JTAG bus 25 where the values can be examined

1 external to the FPGA. The FPGA can be tested in this non-secure mode by using
2 56 bits of a selected 64-bit word in registers 23b as the 56-bit key for DES
3 decryption. In one embodiment, when key memory 23 is in non-secure mode,
4 readback of a user's design is possible even though the design has been encrypted
5 before loading. This allows the designer to test and debug even an encrypted
6 design. Communication of the key security status is through bus 26 (see also
7 Fig. 3).

8 After values have been written into key registers 23b and verified with a
9 read operation from bus 25, control logic 23a is placed into secure mode by using
10 the ISC_PROGRAM_SECURITY instruction and applying logic 0 to bit 0 of the 64-
11 bit key data bus which is part of the IEEE 1532 standard. In the secure mode, no
12 access to the keys is granted.

13 As shown in Fig. 11, to assure that an attacker can not return to the non-
14 secure mode by using the ISC_PROGRAM_SECURITY instruction and then
15 reading out the keys, if the security is eliminated (if the
16 ISC_PROGRAM_SECURITY signal moves to the non-secure logic level), a state
17 machine in control logic 23a erases all keys by writing zeros to all six words, one
18 word at a time. This is done by: in step 110 putting zeros on the wdata[63:0] bus
19 and at step 111 asserting the ws_b[0] signal (with a logic 0 value), then at steps
20 112-117 successively strobing the ws_b[0:0] through ws_b[5:0] signals one at a
21 time before changing the security status at step 118 and entering the non-secure
22 mode, and finally at step 119 releasing the wdata[63:0] logic 0 values. Thus, any
23 attempt to place battery backed up memory 23 into a non-secure mode causes all
24 values in key registers 23b to be erased.

25 To communicate whether key memory 23 is in secure mode, control logic 23a
26 sends a secure mode signal on bus 26 (may be a single line) to configuration logic
27 29 to indicate that key memory 23 is operating in secure mode. If this signal
28 switches to non-secure mode, configuration logic 29 clears the design from
29 configuration memory 12. Note that an unencrypted bitstream may be loaded by

configuration logic 29 into configuration memory 12 even though keys are stored in key registers 23b and key memory 23 is in a secure mode.

Loading the Keys, Multiple Encryption Keys

Decryption keys must be loaded into the PLD before the PLD is put into a secure mode where a user can not learn details of the design. In the embodiment shown in Fig. 3, the key or keys are loaded through a JTAG port 20.

As a feature of the invention, the encryption keys are loaded through this JTAG port 20. It is expected that JTAG programmers will load the encryption keys during board testing. When the RAM for storing keys is in a non-secure mode, the user has full access to it and can read out both the keys and the design, even if the design has been encrypted. This is useful for the designer while testing the keys and the use of the keys. Then once the designer is satisfied with the operation, he or she can send another instruction through the JTAG port and place the key memory into a secure mode. Once the key memory has been placed into secure mode, the keys can not be read out. Further, moving the key memory from secure to non-secure mode erases the keys by activating a circuit that starts up the memory initialization process. (Fig. 15, discussed below, shows a state machine for performing this function.)

According to one aspect of the invention, more than one key may be used to encrypt the design. For example, if three keys are to be used, the bitstream is first encrypted using the first key, then the resulting encrypted bitstream is again encrypted using the second key, then finally the resulting doubly encrypted bitstream is again encrypted using the third key. This triply encrypted bitstream is stored, for example in a PROM or flash memory on the printed circuit board that holds the PLD.

For decryption, these keys are used in succession (reverse order) to repeatedly decrypt the encrypted bitstream. Further to this, if more keys are stored in the PLD than are used for decrypting a particular design, the encrypted bitstream may include in an unencrypted portion an indication of how many keys

1 are to be used, and the address of the first key. Such an embodiment may make it
2 easier for an attacker to decrypt the bitstream because the attacker need only deal
3 with one key at a time. Alternatively, the keys themselves may indicate whether
4 they are the first, middle, last, or only keys. Thus the same PLD can at different
5 times be programmed to perform different functions (configured with different
6 designs), and information about the values of the different keys can be made
7 available to only one or some of the designers. Thus a first designer may not learn
8 about a second design even though both designs are implemented in the same
9 PLD (at different times).

10 Regarding Fig. 3, configuration logic 29 includes additional logic beyond
11 configuration logic 14 of Fig. 1. As in the structure of Fig. 1, the bitstream on
12 configuration access port 21 is treated as words, in one embodiment 32-bit words.
13 Several of the words, usually at or near the beginning of the bitstream, contain
14 header information, for example length of the bitstream, starting address for the
15 configuration data. New to the bitstream of the present invention is an indication
16 as to whether the bitstream is encrypted, and the address of a key for decrypting
17 configuration data in the bitstream.

19 Battery Backed up Memory

20 Values stored in key memory 23 are preferably retained by a battery when
21 power to the FPGA is removed.

22 Further, other memories than encryption keys can also be backed up using a
23 battery supply switch such as switch 22. In particular, a PLD can be
24 manufactured in which the VSWITCH voltage supply is routed to all flip flops in
25 the PLD if the purpose is to preserve data generated by the PLD when the PLD is
26 powered down. And if the purpose is to also preserve configuration of the PLD
27 when the PLD is powered down, configuration memory 12 (Fig. 3) may
28 alternatively be powered from VSWITCH, though such an embodiment requires
29 considerably more battery power than does powering just the flip flops in the

1 PLD, and powering flip flops in turn requires more battery power than does
2 powering a very small memory for storing a few encryption keys.

3 Fig. 12 shows a structure for battery supply switch 22. In this embodiment,
4 VBATT level shift circuit 31 allows the PLD to use different voltages for the
5 battery and main power supply. And of course the purpose of the circuit is to deal
6 with varying voltage levels. In one embodiment, battery supply switch 22 can
7 handle VCCI voltages up to 3.6 volts, and switches to battery power when VCCI
8 falls below about 1 volt. Battery voltage can be between 1.0 volts and 3.6 volts.

9 Battery supply switch 22 includes four output driving P-channel transistors
10 P0 through P3. Transistors P0 and P1 turn on and off together as do transistors P2
11 and P3. The circuit includes two transistors for each leg instead of one in order to
12 avoid any possibility that VCCI and VBATT will be connected together.
13 Transistor P0 includes a parasitic diode (the p-n junction between the drain and
14 substrate) that can conduct current upward in the figure even when the transistor
15 is off. To prevent such current flow, transistor P1 is added and has its substrate
16 connected to its drain so that parasitic diode conduction can only be downward.
17 A similar arrangement is made with transistors P2 and P3. Thus there is no
18 possibility that current will conduct from VBATT to VCCI or from VCCI to
19 VBATT. Inverters 33 and 34 are powered from the VSWITCH voltage, so they are
20 always operational even when VCCI is off. Transistor P4 is a resistor, always on,
21 and provides protection against electrostatic discharge. Most of the time, the
22 structures controlled through transistor P4 do not draw current, so there is usually
23 no voltage drop across transistor P4.

24 Fig. 13 shows one embodiment of VBATT level shift circuit 31. Output
25 voltage at terminal OUT is controlled by signals IN and INB. These signals are
26 generated by inverters 33 and 34, which derive their supply voltage from the
27 VSWITCH node. Therefore, if VSWITCH is supplied by VBATT, one of signals IN
28 and INB will be at voltage VBATT and the other will be at ground. However, if
29 VSWITCH is supplied by VCCI, one of IN and INB will be at the VCCI voltage
30 level. If IN is at VCCI and INB is at ground, transistor 45 will be on and transistor

46 will be off. The gate of P-channel transistor 43 will be low, and transistor 43 will be on, pulling the input of inverter 47 to VBATT. The output of transistor 48 will also be at VBATT. Returning to Fig. 12, a voltage level VBATT at the gate of transistor P0 will positively turn off transistor P0.

Fig. 14 shows VCCI detect circuit 32. VCCI detect circuit 32 determines when the voltage on line VSWITCH will be switched to the battery and back to VCCI. This embodiment of circuit 32 is essentially a string of five inverter stages I1 through I5. Controlling of the switching voltage occurs primarily at inverter stage I1. Transistors 52 and 53 form a CMOS inverter. Power to this CMOS inverter must flow through P-channel transistor 51, which doesn't turn on until VCCI reaches the threshold voltage of transistor 51, typically 0.7-0.8 volts. If VCCI is switching slowly, taking several milliseconds to reach full voltage, transistor 51 delays the activation of circuit I1. When transistor 51 turns on, the source (upper terminal) of transistor 52 goes to VCCI. N-channel transistor 53 typically has a threshold voltage of about 0.7-0.8 volts as well but is sized as a weak transistor relative to transistor 52. In one embodiment, transistor 53 has a width/length ratio of 1/18 whereas transistor 52 has a width/length ratio of 3/2. So transistor 53 pulls the input of inverter I2 low only until transistor 52 turns on. In one embodiment, circuit I1 pulls the input of inverter stage I2 high when VCCI is at about 1.0 volt. Thus the output of inverter 54 goes low. Inverter stage I3 is a Schmitt trigger. The zero volt input to inverter stage I3 turns off transistors 56 and 57 and turns on transistor 55, pulling node N3 to VCCI and turning on transistor 58, which pulls up node N4, thus raising the voltage at which transistor 56 will turn on, and preventing small variations in VCCI from switching the voltage at node N3. Inverters 59 and 60 are optional and produce a sharper edge of the output signals usebatt and usebattb that cause battery supply switch 22 of Fig. 12 to switch from VBATT to VCCI. Transistor 61, controlled by the VBATT' signal, is a weak pull-down transistor and assures that the usebattb line is pulled low when VCCI is not present and therefore not providing an output signal from inverter 60.

Key Not Available to Purchaser of a Product Containing the Configured PLD

In order to prevent an attacker from learning the design that has been used to configure the PLD, several additional steps may be taken.

According to another aspect, a key is loaded into the PLD before sale of a system incorporating the PLD, such that after sale of a system including the PLD, the design can be loaded into the PLD and used, but an attacker can not learn the value stored in the key (or keys). Thus the unencrypted design can not be read or copied. To achieve this security, several steps are taken.

Secure Mode Preservation (Tamper-proofing)

In one embodiment, there are two security flags in configuration logic 29 of the PLD. One indicates whether the decryption keys are secured, and the other indicates whether the design is a decrypted design and must be protected. If JTAG logic 13 (Fig. 3) selects secure mode with the ISC_PROGRAM_SECURITY instruction, a secure_key flag in control logic 23a (Fig. 10a) is set. If the bitstream loaded into the PLD has the indication that design data in the bitstream is encrypted, a secure_design flag in configuration logic 29 (not shown) is set. If either flag is later unset, the entire configuration memory is cleared, thereby removing the decrypted design. If the secure_key flag is reset (by an ISC_PROGRAM_SECURITY instruction), then the keys are also erased.

Fig. 15 shows a state machine for performing the design clearing function. When the secure_design flag is set, the state machine enters state S1. This state monitors a change from secure to non-secure mode of the secure_design flag. As long as the secure-design mode continues, the state machine stays in state S1. Once a change occurs, the state machine enters state S2 and the data shift registers for shifting data into configuration memory 12 are reset, thereby placing zeroes on all data lines for the configuration memory bits. Next, the state machine moves to state S3 where the word line of the addressed frame is asserted. This results in the zeros on the data shift register lines being written into the memory bits at the addressed frame. If question Q1 indicates there are more frames to be addressed,

1 the state machine moves to state S4 where the frame address is advanced and the
2 state machine returns to state S3. When question Q1 indicates there are no more
3 frames to be addressed, the process is done and the configuration memory is
4 cleared.

5 It is also necessary to protect the keys from being accessed by an attacker.
6 Loading of the keys is performed before a system containing the design is made
7 available to an end customer. When designers are in the process of developing the
8 design, they may wish to operate the PLD in a non-secure mode for debugging. In
9 order to allow for this debugging operation and also to preserve security of the
10 keys, the key loading process begins in a non-secure mode by clearing all key
11 registers. A secure key flag must be kept in the non-secure mode while keys are
12 loaded and while the keys are read back for verification. The secure key flag may
13 also be kept in the non-secure mode while a configuration bitstream is loaded and
14 decrypted. But once the secure key flag is set, returning the secure key flag to the
15 non-secure mode clears all keys and also initiates operation of the state machine of
16 Fig. 15. So, not only are the keys cleared, but the configuration is also cleared.

17 Readback Attack and Readback Disabled

18 Some FPGAs allow a bitstream to be read back out of the FPGA so that a
19 user may debug a design or may obtain state machine information from flip flops
20 in the FPGA. Unless the design were re-encrypted for the read-back operation, the
21 act of reading back the bitstream would expose the unencrypted bitstream to
22 view.
23

24 Further security of the design is provided by disabling readback when an
25 encrypted design is loaded into the FPGA. In one embodiment, readback is
26 disabled only if the decryption keys are also secured.

27 Fig. 16 shows the block diagram of a structure for loading and reading back
28 configuration memory. In one embodiment, configuration logic 29 prevents
29 readback when two conditions are present: (1) the security status line on data bus
30 26 (see Figs. 3 and 10) indicates that the keys are in a secure mode, and (2)

configuration logic 29 has responded to op codes in a configuration bitstream that indicate the bitstream is encrypted. So if either the keys are not secured or the bitstream is not encrypted, readback can be enabled. In other embodiments, different conditions control whether readback can be enabled.

When configuration logic 29 receives in the bitstream a header indicating that readback is to be performed, it sends on line 107 the frame address stored in its frame address register, which is decoded by address decoder 110 to select the addressed line of bus 109. Next, word line enable signal on line 108 is asserted, which asserts the selected word line of bus 109 to allow memory cells addressed by the selected word line to place their values on the n data lines 102 (n is the frame length and is stored in configuration logic 29). Configuration logic 29 then asserts the Load signal on line 104 to load the frame of data (in parallel) into data shift register 101. Next, configuration logic 29 asserts the shift signal on line 105 to cause data shift register 101 to shift out the frame of data in 32-bit words on bus 103 to the frame data output register (see Fig. 4d) and from there to an outgoing bitstream on configuration access port 21 (Fig. 3).

If decryption is indicated in the bitstream, configuration logic 29 sets internal flags to indicate this. If these flags are set and key memory 23 is in secure mode as indicated by the security status signal on bus 26, then configuration logic 29 responds to a readback command in the bitstream by keeping the word line enable signal on line 108 inactive and by keeping the load and shift signals on lines 104 and 105 inactive to prevent readback. However, if key memory 23 is not in secure mode, even though the design may be encrypted, readback is allowed so that testing and debugging are possible.

Partial Reconfiguration Attack and Prevention

Some FPGAs allow partial reconfiguration of the FPGA or allow different parts of a design to be loaded into different parts of the FPGA using separate starting addresses and separate write instructions. An attacker might attempt to learn the design by partially reconfiguring the design to read contents of a block

1 RAM or flip flops directly to output ports or by adding a section to an existing
2 design to read out information that can be used to learn the design. For example,
3 the attacker might partially reconfigure the PLD with an unencrypted design
4 whose only purpose is to extract information about the encrypted design. Such a
5 Trojan Horse design could be loaded into the PLD with another bitstream or
6 attached to an existing encrypted bitstream. If the attacker was interested in
7 learning a state machine design loaded into block RAM of an FPGA, for example,
8 the Trojan Horse design could include logic to cycle through the addresses of the
9 block RAM and send the block RAM data contents to package pins.

10 In order to prevent an attacker from making such changes, if the original
11 design is encrypted, configuration logic 29 disallows partial reconfiguration once
12 configuration with decryption is started. Configuration logic 29 disallows a
13 further write instruction once a header with the decryption op code has been
14 processed. Also, configuration logic 29 disallows configuration with decryption
15 once configuration without encryption has been done. Configuration logic 29
16 accomplishes these restrictions by ignoring headers that write to configuration
17 memory after a decrypt instruction has been received and ignoring headers that
18 have a decrypt command if an unencrypted portion of a design has been loaded.
19 Thus, if any op code indicates that writing with decryption is being used, the PLD
20 will accept only a single write instruction.

21 Additional Embodiments

22 The above description of the drawings gives detail on a few embodiments.
23 However, many additional embodiments are also possible. For example, instead
24 of the cipher block chaining algorithm discussed above, one can use an encryption
25 method called cipher feedback mode in which data can be encrypted in units
26 smaller than the block size, for example one 8-bit byte at a time. This cipher-
27 feedback mode is described by Schneier, *ibid*, at pages 200-203.

28 In yet another embodiment, if encryption is used, all bitstreams must be
29 loaded starting at address 0. One implementation of this embodiment replaces
30

1 any address loaded into the starting frame address register FAR (Fig. 6) with
2 address 0 when an op code specifying encryption is received.

3 In still another embodiment, the starting address and the design data are
4 both encrypted. In this embodiment, it is possible to load several segments of
5 encrypted design data starting at different frame addresses, just as is possible with
6 unencrypted design data.

7 In another embodiment, the key data stored in a key memory such as key
8 memory 23 specifies the number of keys that will follow. In a variation on this
9 embodiment, the key data also specify the number of keys that precede the key. If
10 an attacker gives a key address other than the first key address intended by the
11 designer, the configuration may be aborted. Additionally, encryption will proceed
12 until the number of keys specified within the keys have been used.

13 In another embodiment, instead of allowing keys to be read back when the
14 key memory is in a non-secure mode, the keys include parity bits or CRC check
15 bits, and only these bits can be read back for verification that the key or keys were
16 loaded correctly. This embodiment allows keys known to one designer to be kept
17 secret from another designer, and is useful when the PLD is to be used at different
18 times for loading different designs.

19 Regarding the CRC checksum calculation discussed above, embodiments can
20 be provided in which the CRC checksum is calculated either before or after a
21 design is encrypted. Of course, if the checksum added to the bitstream is
22 calculated before the design data is encrypted, then a corresponding checksum
23 must be calculated within the PLD on the design data after it has been decrypted.
24 Likewise, if the checksum added to the bitstream is calculated after the design
25 data has been encrypted, then the PLD must calculate the corresponding
26 checksum on the received bitstream before the design data have been decrypted.

27 A further note regarding the process of loading the decryption keys, when
28 the process illustrated in Fig. 8 is used, it is not necessary to use a device
29 programmer for loading decryption keys. The keys may simply be loaded as part
30 of the board test procedure.

1 It is also possible to use the structures and methods described above for
2 programming more than one PLD. It is well known to use a single bitstream for
3 programming more than one PLD or FPGA, either by arranging several devices in
4 a daisy chain and passing the bitstream through the devices in series or
5 addressing the devices in series. It is possible to arrange several PLDs in such an
6 arrangement when one or more of the devices is to receive encrypted design data.

7 As yet another embodiment, although one embodiment was described in
8 which only a single address could be specified for a bitstream having encrypted
9 design data, in another embodiment, several addresses, preferably encrypted, can
10 be specified for loading separate portions of a design. Further, these separate
11 portions may use the same encryption key or keys, or the separate portions may
12 use different encryption keys or different sets of keys.

13 Variations that have become obvious from the above description are
14 intended to be included in the scope of the invention.

CLAIMS

1. A PLD with decryption and structure for preventing design relocation comprising:
 - a decryptor for decrypting an encrypted bitstream;
 - an address indicator for indicating an address into which configuration data will be loaded; and
 - a decryption algorithm implemented by the decryptor, wherein the decryption algorithm uses data from the address indicator for decrypting the encrypted bitstream.
2. The PLD of Claim 1 wherein the address indicator is an initial address indicator.
3. The PLD of Claim 2 wherein the initial address indicator is a frame address for indicating a starting frame of the PLD into which configuration data will be loaded.
4. The PLD of Claim 1 wherein the decryption algorithm comprises the DES algorithm.
5. The PLD of Claim 4 wherein the DES algorithm includes a cipher block chaining algorithm and the address indicator is placed into a starter value of the cipher block chaining algorithm.
6. The PLD of Claim 4 wherein the DES algorithm includes a cipher feedback mode algorithm and the address indicator is placed into a starter value of the cipher block chaining algorithm.
7. The PLD of Claim 1 wherein the address indicator specifies a constant value.

- 1 8. The PLD of Claim 1 wherein the address indicator is loaded with a value in the
- 2 bitstream.
- 3
- 4 9. The PLD of Claim 8 wherein the value in the bitstream is encrypted.
- 5
- 6 10. The PLD of Claim 8 wherein the value in the bitstream is not encrypted.

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212

1 PROGRAMMABLE LOGIC DEVICE WITH DECRYPTION AND
2 STRUCTURE FOR PREVENTING DESIGN RELOCATION

3
4 Stephen M. Trimberger
5 Raymond C. Pang
6 Walter N. Sze
7 Jennifer Wong
8
9

10 It is sometimes desirable to protect a design used in a PLD from being
11 copied. According to the present invention, the design is encrypted, then loaded
12 into a PLD, then decrypted, and then loaded into the configuration memory of the
13 PLD. An attacker could relocate the design to a visible part of the PLD and learn
14 the design. The present invention prevents design relocation by attaching address
15 information to the encryption key or by encrypting an address where the design is
16 to be loaded as well as encrypting the design itself. Thus, if an attacker tries to
17 load the design into a different part of the PLD, the encrypted design will not
18 decrypt properly.

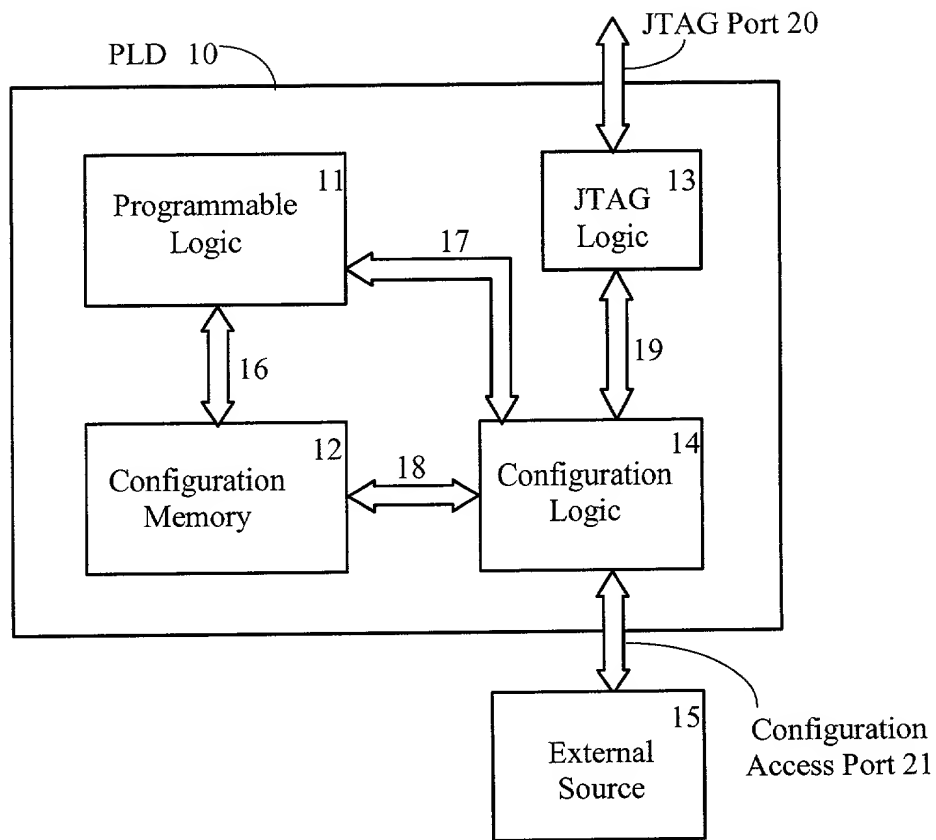


Fig. 1
PRIOR ART

Header A
Data A
Header B
Data 1B
Data 2B
Data 3B
Data 4B
Header C
Data 1C
⋮

Fig. 2a
PRIOR ART

Bitstream header Config. Logic Reg. address	Config. Logic Register data contents
0000	Cyclic Redundancy Check (CRC)
0001	Frame Address
0010	Frame Data Input
0011	Frame Data Output
0100	Command
0110	Control
0111	Status
1000	Daisy Chain Output
1001	Configuration Option
1010	Reserved
1011	Frame Length
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Fig. 2d
PRIOR ART

32-bit Command Header Format 1

Format Type	Op code	Config. Logic Register address	Word count
001	00 01 10		

Fig. 2b PRIOR ART

11 bits, counts 2048 words

32-bit Command Header Format 1

Format Type	Op code	Config. Logic Register Address	Word Count
001	00 01 10		000000000000

32-bit Command Header Format 2

Format Type	Op code	Word Count
010	00 01 10	

27 bits, counts 2.68 million words or 10 megabytes

Fig. 2c PRIOR ART

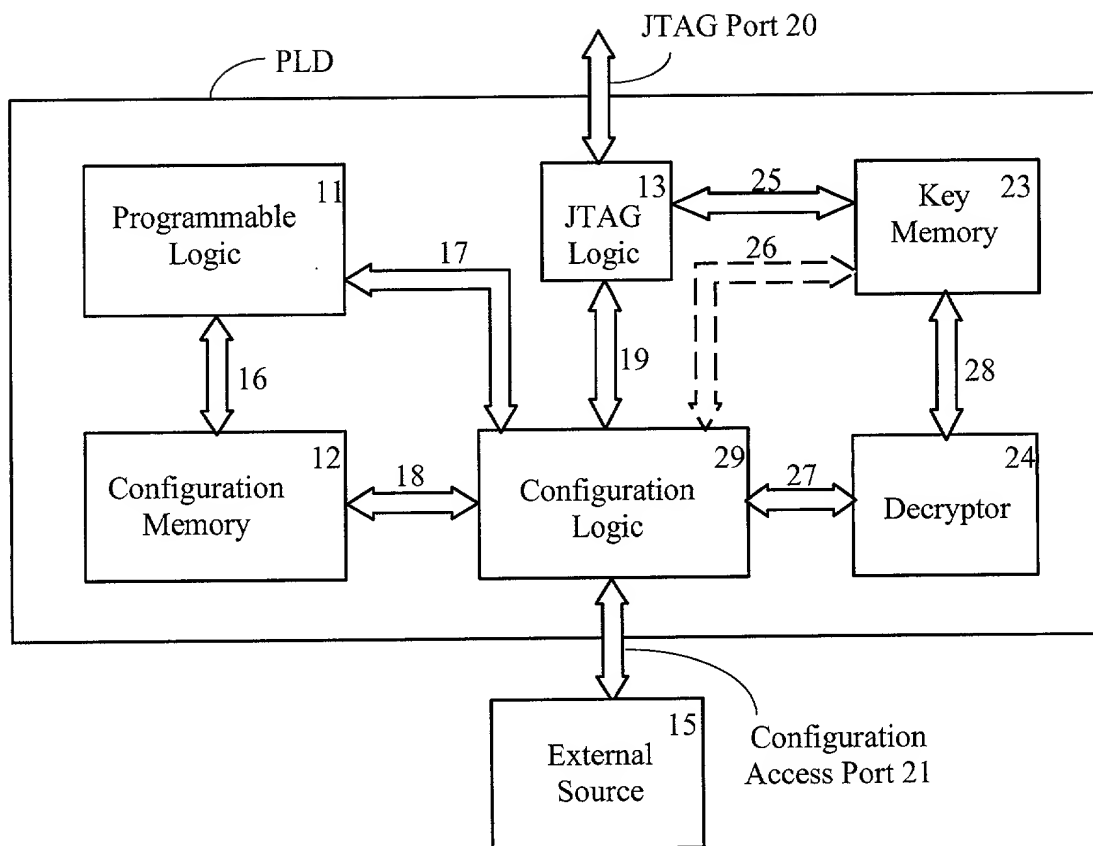


Fig. 3

Header A
Data A
Header B
Data B
Header C
Data C
Header D
Data 1D
Data 2D
Data 3D
⋮

Fig. 4a

Bitstream header Config. Logic Reg. address	Config. Logic Register data contents
0000	Cyclic Redundancy Check (CRC)
0001	Frame Address
0010	Frame Data Input
0011	Frame Data Output
0100	Command
0110	Control
0111	Status
1000	Daisy Chain Output
1001	Configuration Option
1010	Reserved
1011	Frame Length
1100	Cipher block chaining(CBC) starter
1101	Initial key address
1110	Reserved
1111	Reserved

Fig. 4d

32-bit Command Header Format 1

Format Type	Op code	Config. Logic Register address	Word count
001	00 01 10 11		

Fig. 4b

11 bits, counts 2048 words

32-bit Command Header Format 1

Format Type	Op code	Config. Logic Register Address	Word Count
001	00 01 10 11		000000000000

32-bit Command Header Format 2

Format Type	Op code	Word Count
010	00 01 10 11	

Fig. 4c

27 bits, counts 2.68 million words or 10 megabytes

```

Dummy word      111111111111111111111111111111111111
Sync word      1010101010011001010101010101010110
Type 1 write 1 words to CMD 0011000000000000010000000000000001
RCRC command   0000000000000000000000000000000111
Type 1 write 1 words to FLR 0011000000000000010100000000000001
data word 0    0000000000000000000000000000011001
Type 1 write 1 words to COR 0011000000000000100100000000000001
data word 0    000000000000001000011111111000001
Type 1 write 1 words to ID  0011000000000000111000000000000001
data word 0    00000001000000000100000001001001
Type 1 write 1 words to MASK 0011000000000000011000000000000001
data word 0    0000000000000000000000000000000000
Type 1 write 1 words to CMD 0011000000000000010000000000000001
SWITCH command 000000000000000000000000000001001
Type 1 write 1 words to FAR 0011000000000000001000000000000001
data word 0    0000000000000000000000000000000000
Type 1 write 1 words to CMD 0011000000000000010000000000000001
WCFG command   0000000000000000000000000000000001
Type 1 write 0 words to FDRI 0011000000000000010000000000000000
Type 2 write 10530 words to FDRI 01010000000000000010100100100010
data word 0    0000000000000000000000000000000000
data word 1    0000000000000000000000000000000000
data word 2    0000100000000000000000000000000000
data word 3    0000100000000000011111111111111100
:
:
:
data word 10526 0000000000000000000000000000000000
data word 10527 0000000000100000000000000000000001
data word 10528 0000000000000000000000000000000000
data word 10529 0000000000000000000000000000000000
Auto CRC word  0000000000000000000110111000010001
Type 1 write 1 words to CMD 0011000000000000010000000000000001
LFRM command   0000000000000000000000000000000001
Type 1 NO OP   0010000000000000000000000000000000
:
:
:
Type 1 NO OP   0010000000000000000000000000000000
Type 1 write 1 words to COR 0011000000000000100100000000000001
data word 0    00000001000001000000111111000001
Type 1 write 1 words to CMD 0011000000000000010000000000000001
START command  000000000000000000000000000000000101
Type 1 write 1 words to CTL 0011000000000000010100000000000001
data word 0    0000000000000000000000000000000000
Type 1 write 1 words to CMD 0011000000000000010000000000000001
GRESTORE command 00000000000000000000000000000001010
Type 1 write 1 words to CRC 0011000000000000000000000000000001
data word 0    00000000000000000000010100100110010
Type 1 write 1 words to CMD 0011000000000000010000000000000001
DESYNCH command 0000000000000000000000000000000001101
Type 1 NO OP   0010000000000000000000000000000000

```

Parameter	Unit	Value	Standard Error	t-value	p-value
Intercept		1.000	0.000	1.000	0.000
Age	Years	0.000	0.000	0.000	0.000
Gender		0.000	0.000	0.000	0.000
Marital Status		0.000	0.000	0.000	0.000
Education	Years	0.000	0.000	0.000	0.000
Income	\$/Year	0.000	0.000	0.000	0.000
Health		0.000	0.000	0.000	0.000
Religion		0.000	0.000	0.000	0.000
Occupation		0.000	0.000	0.000	0.000
Region		0.000	0.000	0.000	0.000
Time	Years	0.000	0.000	0.000	0.000
Time Squared		0.000	0.000	0.000	0.000
Time Cubed		0.000	0.000	0.000	0.000
Time Quart		0.000	0.000	0.000	0.000
Time Quint		0.000	0.000	0.000	0.000
Time Sext		0.000	0.000	0.000	0.000
Time Sept		0.000	0.000	0.000	0.000
Time Oct		0.000	0.000	0.000	0.000
Time Eleventh		0.000	0.000	0.000	0.000
Time Twelfth		0.000	0.000	0.000	0.000
Time Thirteenth		0.000	0.000	0.000	0.000
Time Fourteenth		0.000	0.000	0.000	0.000
Time Fifteenth		0.000	0.000	0.000	0.000
Time Sixteenth		0.000	0.000	0.000	0.000
Time Seventeenth		0.000	0.000	0.000	0.000
Time Eighteenth		0.000	0.000	0.000	0.000
Time Nineteenth		0.000	0.000	0.000	0.000
Time Twentieth		0.000	0.000	0.000	0.000
Time Twenty-first		0.000	0.000	0.000	0.000
Time Twenty-second		0.000	0.000	0.000	0.000
Time Twenty-third		0.000	0.000	0.000	0.000
Time Twenty-fourth		0.000	0.000	0.000	0.000
Time Twenty-fifth		0.000	0.000	0.000	0.000
Time Twenty-sixth		0.000	0.000	0.000	0.000
Time Twenty-seventh		0.000	0.000	0.000	0.000
Time Twenty-eighth		0.000	0.000	0.000	0.000
Time Twenty-ninth		0.000	0.000	0.000	0.000
Time Thirtieth		0.000	0.000	0.000	0.000
Time Thirty-first		0.000	0.000	0.000	0.000
Time Thirty-second		0.000	0.000	0.000	0.000
Time Thirty-third		0.000	0.000	0.000	0.000
Time Thirty-fourth		0.000	0.000	0.000	0.000
Time Thirty-fifth		0.000	0.000	0.000	0.000
Time Thirty-sixth		0.000	0.000	0.000	0.000
Time Thirty-seventh		0.000	0.000	0.000	0.000
Time Thirty-eighth		0.000	0.000	0.000	0.000
Time Thirty-ninth		0.000	0.000	0.000	0.000
Time Fortieth		0.000	0.000	0.000	0.000
Time Forty-first		0.000	0.000	0.000	0.000
Time Forty-second		0.000	0.000	0.000	0.000
Time Forty-third		0.000	0.000	0.000	0.000
Time Forty-fourth		0.000	0.000	0.000	0.000
Time Forty-fifth		0.000	0.000	0.000	0.000
Time Forty-sixth		0.000	0.000	0.000	0.000
Time Forty-seventh		0.000	0.000	0.000	0.000
Time Forty-eighth		0.000	0.000	0.000	0.000
Time Forty-ninth		0.000	0.000	0.000	0.000
Time Fiftieth		0.000	0.000	0.000	0.000
Time Fifty-first		0.000	0.000	0.000	0.000
Time Fifty-second		0.000	0.000	0.000	0.000
Time Fifty-third		0.000	0.000	0.000	0.000
Time Fifty-fourth		0.000	0.000	0.000	0

 ENCRYPTED VERSION

```

      Dummy word 11111111111111111111111111111111
      Sync word 1010101010011001010101010101100110
Type 1 write 1 words to CMD 00110000000000001000000000000001
      RCRC command 0000000000000000000000000000000111
Type 1 write 1 words to FLR 0011000000000000101100000000000001
      data word 0 000000000000000000000000000000011001
Type 1 write 1 words to COR 0011000000000000100100000000000001
      data word 0 00000000000000001000011111111000001
Type 1 write 1 words to ID 0011000000000000111000000000000001
      data word 0 00000001000000001000000010010011
Type 1 write 1 words to MASK 0011000000000000011000000000000001
      data word 0 0000000000000000000000000000000000
Type 1 write 1 words to CMD 0011000000000000100000000000000001
      SWITCH command 00000000000000000000000000000001001
Type 1 write 1 words to FAR 0011000000000000001000000000000001
      data word 0 0000000000000000000000000000000000
Type 1 write 1 words to CMD 0011000000000000010000000000000001
      WCFG command 0000000000000000000000000000000001
Type 1 write 1 word to KEY ADDR 0011000000000000110000000000000001
      data word 0 0000000000000000000000000000000000
Type 1 write 2 words to CBC 0011000000000000110100000000000010
      data word 0 100100001010101111100110111101111
      data word 1 00010010001101000101011001111000
Type 1 decrypt 0 words to FDRI 00111000000000000100000000000000
Type 2 dec 10530 words to FDRI 01011000000000000010100100100010
      data word 0 00010101110110110011001011001000
      data word 1 111000001110001101111111101100111
      data word 2 10101010011111011110001100110011
      data word 3 01100001101011100001011010110100
      .
      .
      .
data word 10526 010110011011111101100010101001010
data word 10527 00101001010111111101011110110000
data word 10528 11000000001010100110011000101010
data word 10529 01111111110010000001100011100010
      Auto CRC word 000000000000000001011111011001010
Type 1 write 1 words to CMD 0011000000000000010000000000000001
      LFRM command 0000000000000000000000000000000011
      Type 1 NO OP 0010000000000000000000000000000000
      .
      .
      .
      Type 1 NO OP 0010000000000000000000000000000000
Type 1 write 1 words to COR 0011000000000000100100000000000001
      data word 0 00000000000000001000011111111000001
Type 1 write 1 words to CMD 0011000000000000010000000000000001
      START command 0000000000000000000000000000000101
Type 1 write 1 words to CTL 0011000000000000010100000000000001
      data word 0 0000000000000000000000000000000000
Type 1 write 1 words to CMD 0011000000000000010000000000000001
      GRESTORE command 00000000000000000000000000000001010
Type 1 write 1 words to CRC 0011000000000000000000000000000001
      data word 0 00000000000000000000001010111001111
Type 1 write 1 words to CMD 0011000000000000010000000000000001
      DESYNCH command 000000000000000000000000000000001101
      Type 1 NO OP 0010000000000000000000000000000000

```

Fig. 5b

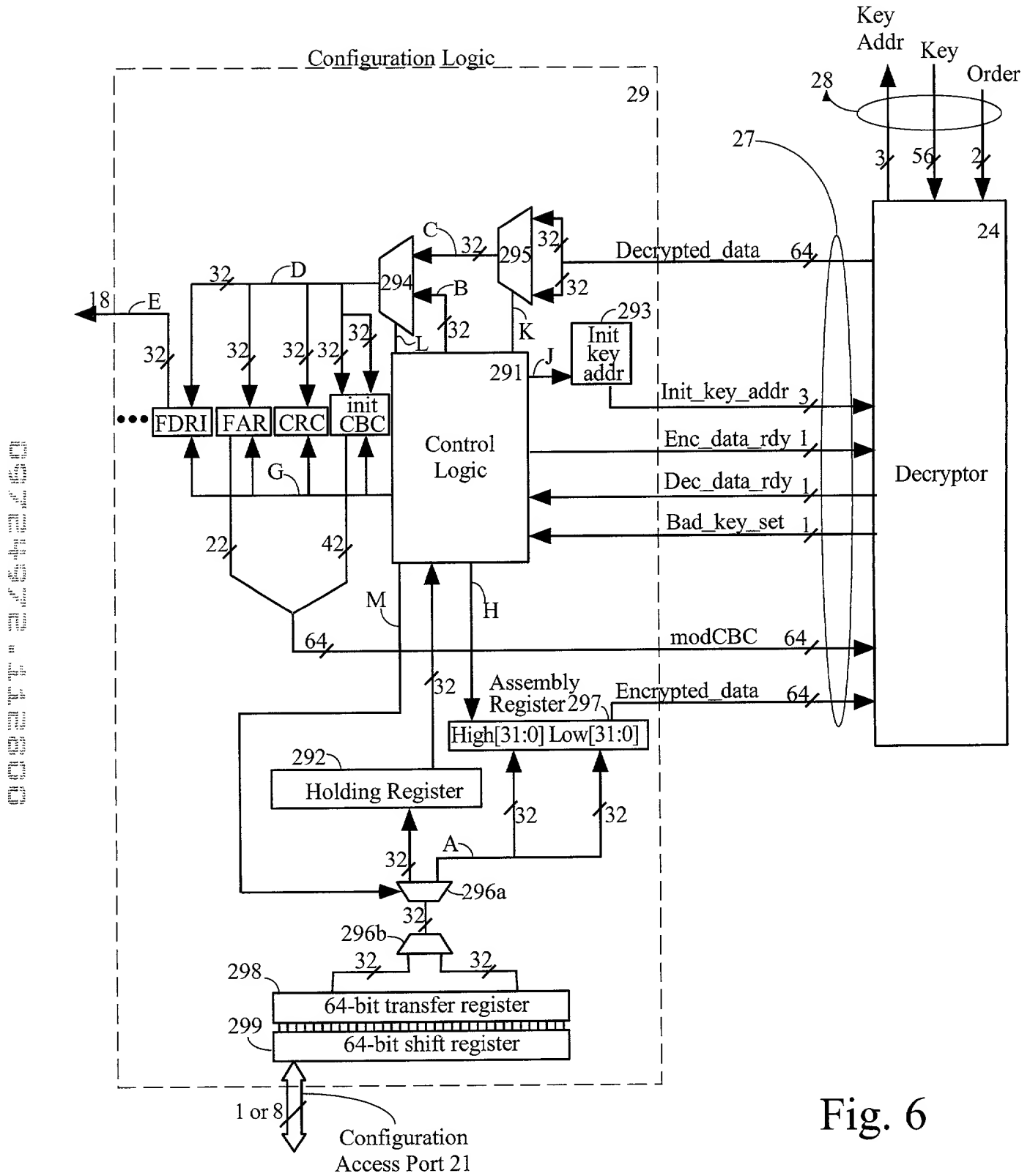


Fig. 6

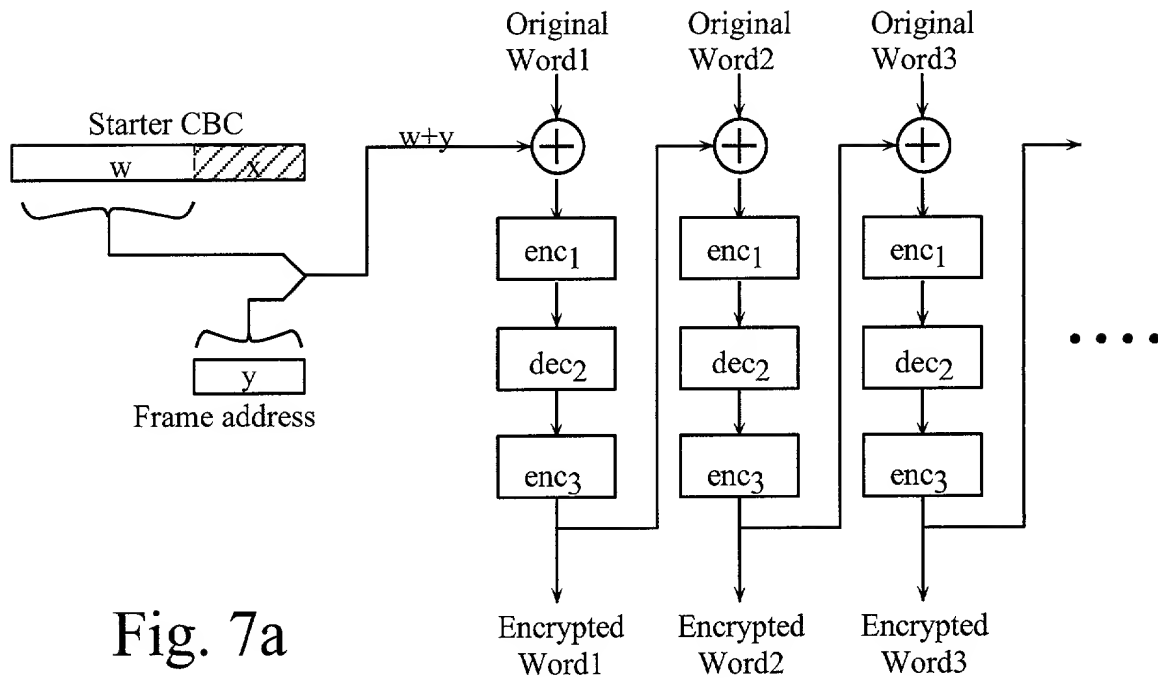


Fig. 7a

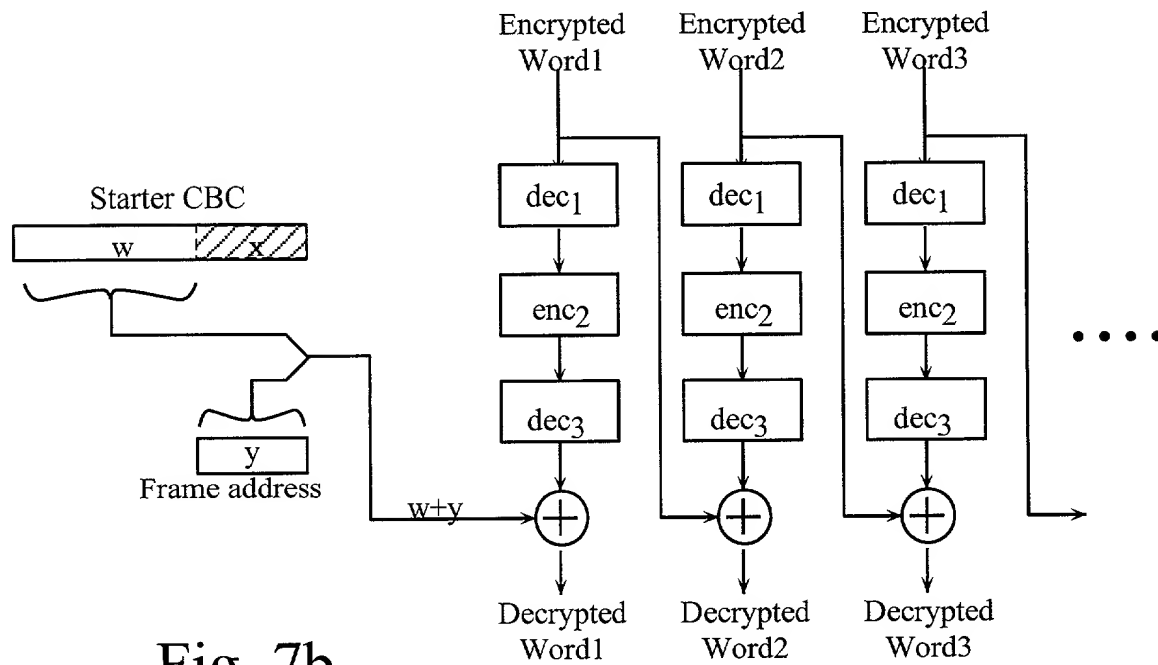


Fig. 7b

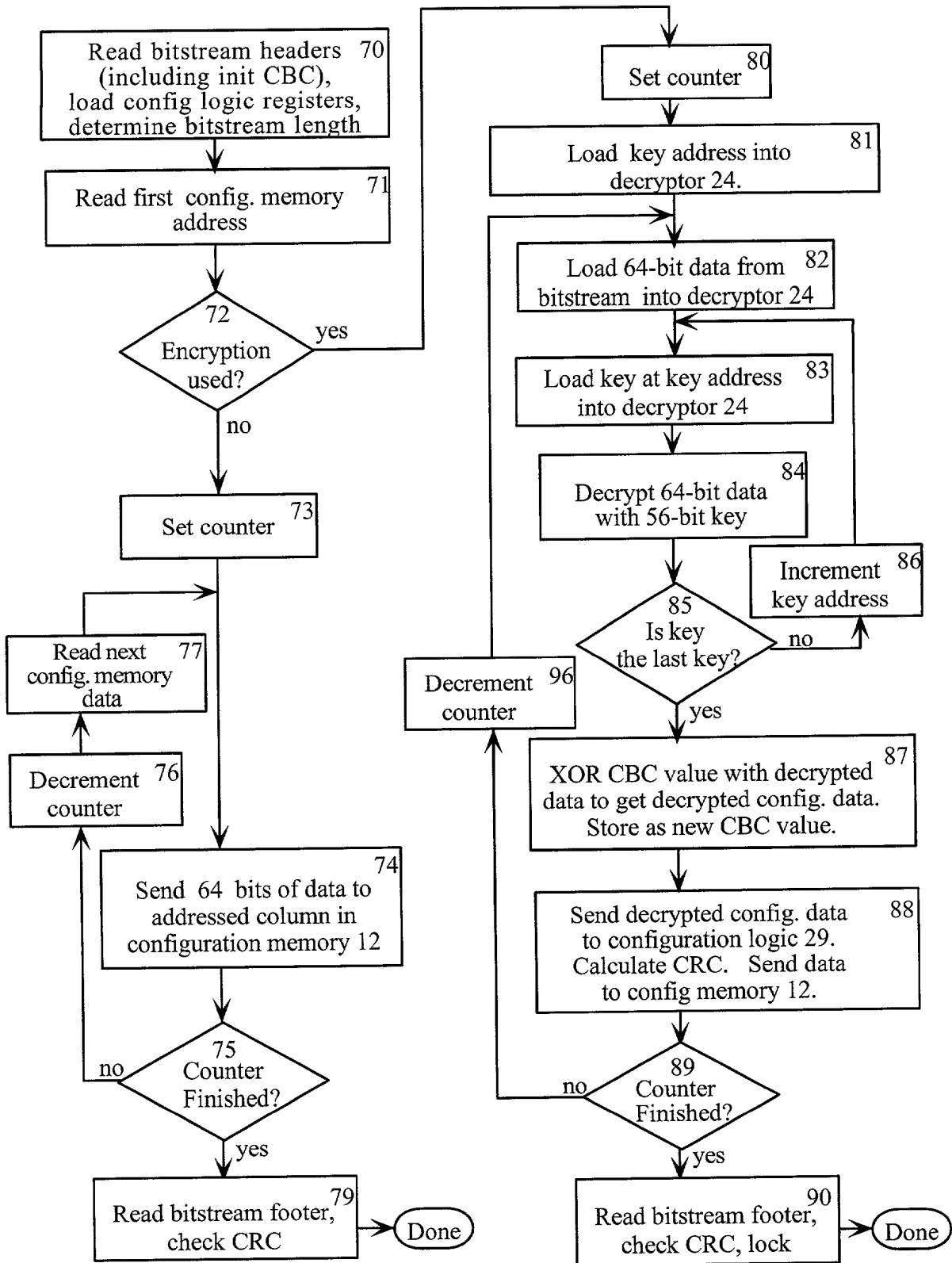


Fig. 8

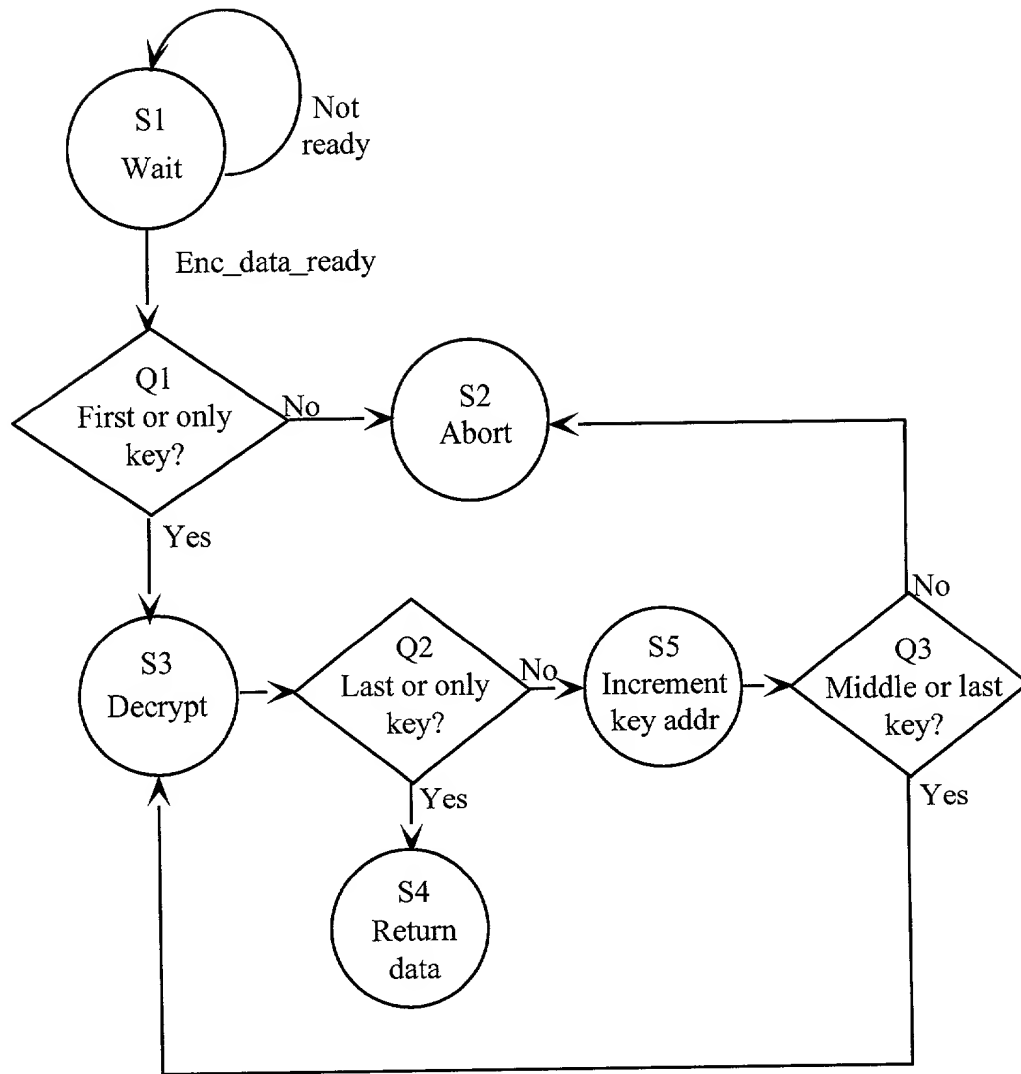


Fig. 9

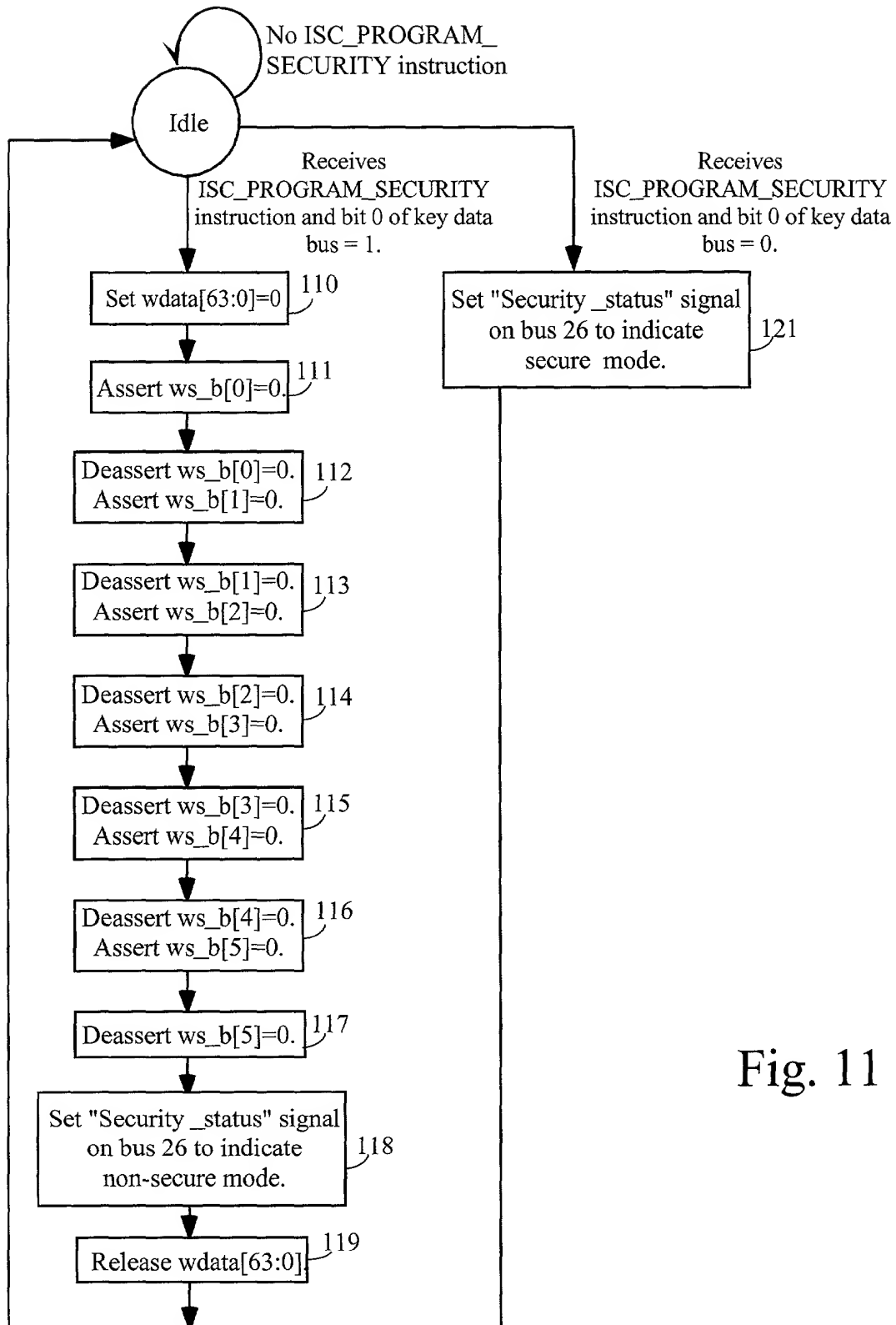
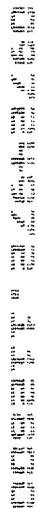


Fig. 11



1. *Chlorophyll a* (Chl *a*) is the primary photosynthetic pigment in most plants and algae. It is a green pigment that absorbs light energy in the blue and red regions of the visible spectrum.

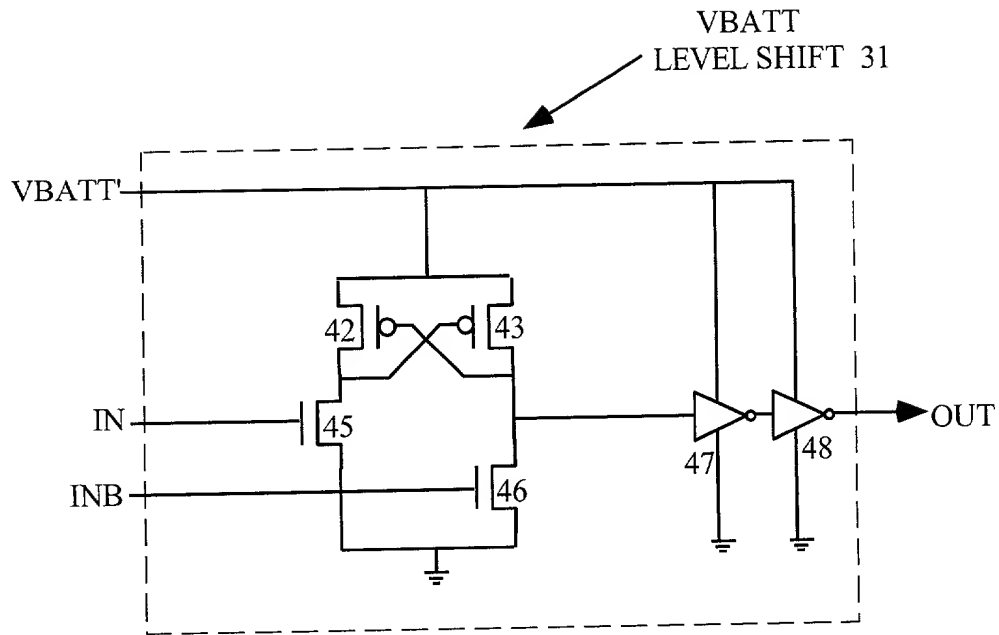


Fig. 13

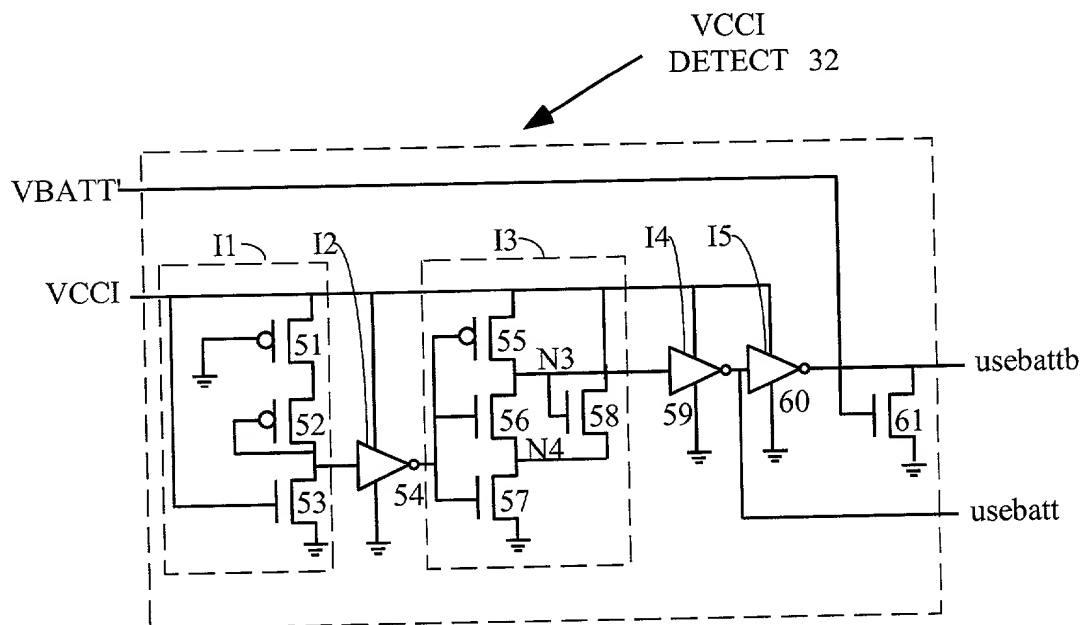


Fig. 14

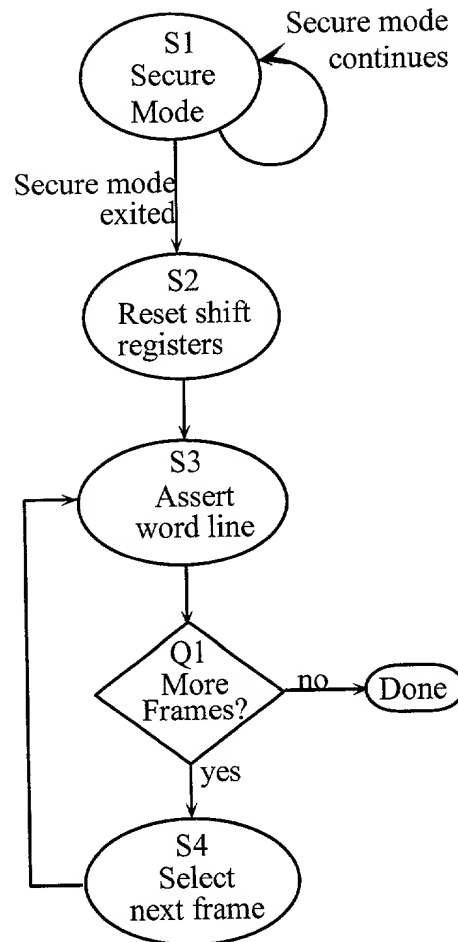


Fig. 15

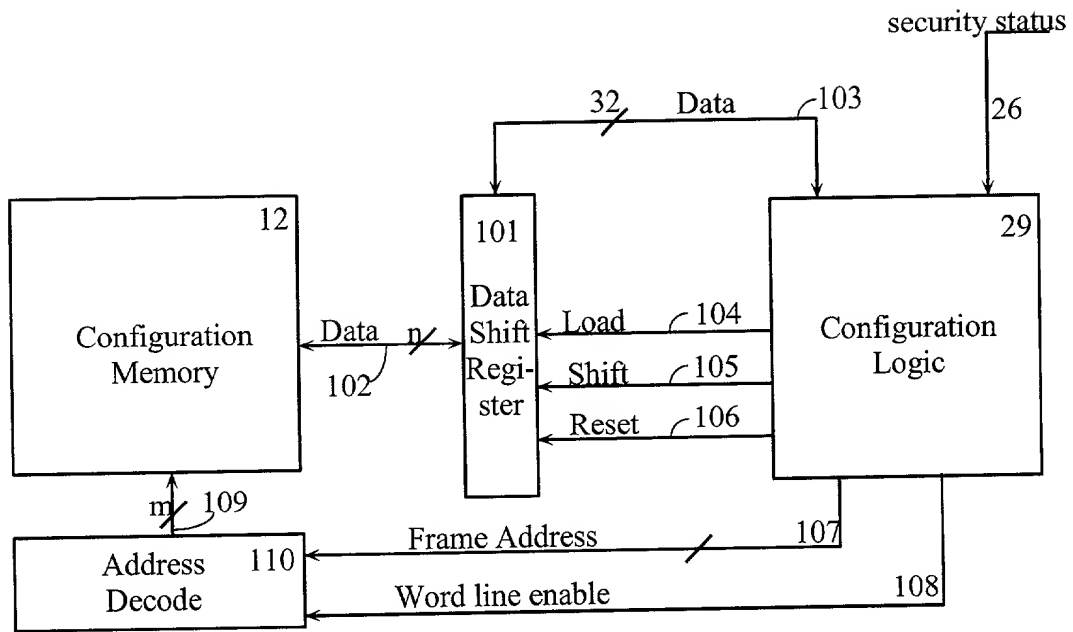


Fig. 16

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) that is disclosed and/or claimed and for which a patent is solicited by way of the application entitled

PROGRAMMABLE LOGIC DEVICE WITH DECRYPTION AND STRUCTURE FOR PREVENTING DESIGN RELOCATION

which (check)

- ☒ is attached hereto.
☐ and is amended by the Preliminary Amendment attached thereto.
☐ was filed on _____ as Application Serial No. _____.
☐ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified application, including the claims, including portions amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).

I hereby claim foreign priority benefits under title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)	Priority Claimed	
	Yes	No
_____ (Number) (Country) (Day/Month/Year Filed)		
_____ (Number) (Country) (Day/Month/Year Filed)		

I hereby claim the benefit under 35 U.S.C. § 119(e) of any United States provisional application(s) listed below

_____ (Application Number(s))	_____ (Filing Date (MM/DD/YYYY))
----------------------------------	-------------------------------------

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as any subject matter of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56(a) which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application:

_____ (Application Serial No.)	_____ (Filing Date)	_____ (Status-patented, pending, abandoned)
_____ (Application Serial No.)	_____ (Filing Date)	_____ (Status-patented, pending, abandoned)

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected herewith:

Keith A. Chanroo (36,480), Edel M. Young (32,451), Lois D. Cartier (40,941), and H. C. Chan (35,477) all located at Xilinx, Inc., 2100 Logic Drive, San Jose, California 95124.

Address all telephone calls to: Edel M. Young at Telephone No. (408) 879-4969
Address all correspondence to: Edel M. Young
Xilinx, Inc.
2100 Logic Drive
San Jose, California 95124

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, § 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor Stephen M. Trimberger

Inventor's signature *Stephen M. Trimberger* Date 11/28/00

Residence 1261 Chateau Drive, San Jose, CA 95120 Citizenship United States

Post Office Address 1261 Chateau Drive, San Jose, CA 95120

Full name second inventor Raymond C. Pang

Inventor's signature *Raymond C. Pang* Date 11-28-00

Residence 1138 Falcon Ridge Court, San Jose, CA 95120 Citizenship United States

Post Office Address 1138 Falcon Ridge Court, San Jose, CA 95120

Full name third inventor Walter N. Sze

Inventor's signature *Walter N. Sze* Date 11/28/00

Residence 20439 Kirkmont Drive, Saratoga, CA 95070 Citizenship United States

Post Office Address 20439 Kirkmont Drive, Saratoga, CA 95070

Full name fourth inventor Jennifer Wong

Inventor's signature *Jennifer Wong* Date 11/28/00

Residence 40565 Encanto Way, Fremont, CA 94539 Citizenship United States

Post Office Address 40565 Encanto Way, Fremont, CA 94539

EMY:mom